NEW ANALYTIC MODELS FOR MULTIPROCESSORS WITH VARIOUS INTERCONNECTION STRUCTURES

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

By
ADARSHPAL SINGH SETHI

[illegible]

to the
COMPUTER SCIENCE PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
AUGUST, 1977

CENTRAL LIBRARY
Acc. No. 54882
19 AUG 1978

To

Brajesh and Prabha

POST GRADUATE OFFICE
I.I.T Kanpur.
Submitted on 23.8.1977
BR

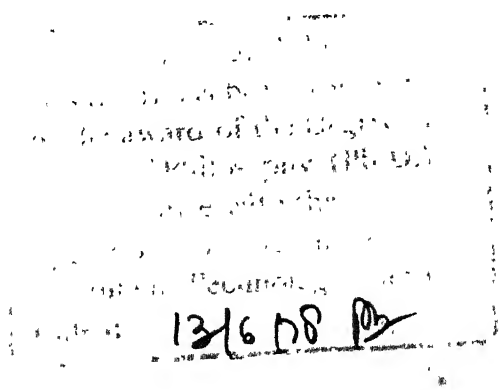
CERTIFICATE

This is to certify that the work entitled "NEW ANALYTIC MODELS FOR MULTIPROCESSORS WITH VARIOUS INTERCONNECTION STRUCTURES" by Adarshpal Singh Sethi has been carried out under my supervision and has not been submitted elsewhere for a degree.

Narsingh Deo

Kanpur
August 16, 1977

Narsingh Deo
Professor
Computer Science Programme
Indian Institute of Technology
Kanpur



ACKNOWLEDGEMENTS

I am deeply indebted to my thesis advisor Professor Narsingh Deo who has been a source of constant inspiration throughout the period of this work. His inspiring guidance, advice, and constructive criticisms were invaluable towards the completion of this thesis and are acknowledged with gratitude.

I am thankful to Professor V. Rajaraman for his encouragement and the abiding interest he showed in my work. I wish to thank Dr. M.S. Krishnamoorthy on whose time I always had a pre-emptive priority and who was ever ready with valuable suggestions. I am grateful to my friends Mr. B.H. Jajoo and Mr. B.K. Gairola for the many fruitful discussions I had with them.

Special thanks are due to Dr. D.P. Bhandarkar of Texas Instruments, Dallas, U.S.A., whose suggestions have greatly improved the material contained in Chapter 3.

I also wish to express my thanks to Mr. H.K. Nathani who has taken great pains to do an excellent job of the typing of the manuscript.

Kanpur
August 16, 1977

- A.S. Sethi

CONTENTS

LIST OF FIGURES

-

LIST OF TABLES

SYNOPSIS

of the

Ph. D. Dissertation

on

NEW ANALYTIC MODELS FOR MULTIPROCESSORS WITH VARIOUS INTERCONNECTION STRUCTURES

by

Adarshpal Singh Sethi

Computer Science Programme
Indian Institute of Technology, Kanpur

August 1977

With the rapid evolution of computer technology has come the need to configure cheap computer systems with large processing power and high reliability. One method of achieving these goals has been to exploit the parallelism of multiprocessor systems. In recent years, an increasing number of multiprocessors have been designed and/or built, such as C.mmp at Carnegie-Mellon University, the TDC-316 multiprocessor at the Tata Institute of Fundamental Research, Bombay, and the BBN Pluribus Interface Message Processor for the ARPA Network. This activity has given a spurt to work on computer modelling to analyse the performance of such systems at the level of the processor-memory interface.

Work on performance evaluation, however, still lags far behind the advances in multiprocessor technology. The performance of a multiprocessor system is crucially dependent on the interconnection mechanism

used for communication between the functional units. Hence the prime effert of ongoing research in this area is to devise better and more efficient interconnection schemes. The system designer must then have adequate tools available to enable him to evaluate and compare the performances of multiprocessors which use these various schemes. It is the problem of devising such evaluation techniques that we address ourselves to in this thesis.

We start the thesis by describing some of the important interconnection schemes being used in multiprocessors. These are the crossbar switch, time-shared bus, multiport memory/multibus, and a hybrid interconnection scheme used in the Pluribus multiprocessor built by BBN for the ARPA Network. Salient characteristics as well as the advantages and disadvantages of these schemes are discussed.

We next give a summary of the existing work on analytic models for the crossbar switch multiprocessors. Most of the past research on this topic has assumed that the memory references of each processor are uniformly distributed among all the memory modules. Although this assumption considerably simplifies the analysis, it is not realistic, since programs generally exhibit the property of locality of references.

The first new result in this thesis is the development of a model for crossbar switch multiprocessors with local referencing, which reflects more closely the behavior of real systems. This model is analysed using both discrete and continuous Markov chain techniques, and expressions are derived for the multiprocessor performance. New expressions are also obtained for the performance in the traditional uniform

reference model and are compared with other expressions available in the literature. Results of a simulation study are presented to demonstrate the accuracy of the expressions for both models.

Almost all the work to date on computer modelling for analysing the performance of multiprocessor systems has been limited to the study of systems using a crossbar switch as the interconnection medium. As mentioned earlier, the tools of analytic modelling need to be improved to keep pace with the innovative development of new interconnection schemes. One of the main contributions of this thesis is the construction of analytic models for multiprocessors using the time-shared bus and the hybrid Pluribus scheme as the interconnection structures.

A discrete Markov chain model for time-shared bus multiprocessors is described. An example is given to explain the detailed analysis technique and simulation results are presented to verify the results of the analysis.

Next, a model for evaluating the performance of the Pluribus multiprocessor is described. The Pluribus system is a hybrid containing a crossbar switch and a number of time-shared buses. The analytic model described here breaks the system into its crossbar switch and time-shared bus components, simultaneously taking into account the complex interaction between these components. The crossbar switch is then analysed in terms of an existing model while the time-shared bus component is analysed using the model developed earlier. These results are synthesized to give the performance of the whole system. Graphical results are presented to show the effect of the various parameters of

the system on its performance. Simulation results presented validate the model.

Finally, some suggestions are made for further work in this area.

CHAPTER 1

INTRODUCTION

With the rapid evolution of computer technology has come the need to construct computer systems which will solve larger problems in less time with higher reliability. Parallel processing represents one of the more effective ways of achieving these goals. The initial development of systems incorporating parallelism was almost entirely motivated by considerations of reliability. Thus redundancy was provided at various levels of the system to cater for catastrophic failure situations.

It gradually came to be realized, however, that redundant components could actually be put to use to improve the performance of the system. If some of the resources fail, dynamic re-allocation of the remaining resources then results in graceful degradation or "fail-soft" operation.

Another advantage of several parallel systems is their flexibility. Flexibility, in the words of Searle and Freberg [Sea 75] "is a measure of the ease with which a system configuration can be altered." For a system to be truly flexible, both its hardware and software should be capable of easy alteration. A better understanding of operating systems for large parallel systems has emerged recently, thus allowing them to be made more flexible. In fact, availability of better software is one of the factors responsible for the increasing popularity of such systems.

However, the greatest potential benefit of parallel systems lies in their performance capabilities. Electronic technology appears to be

approaching limits imposed by electrical propagation delays. Parallel processing offers an attractive means of overcoming this problem. Plummetting hardware costs have also improved the cost/performance viability of these systems. The future is thus certainly going to witness an increasing exploitation of concurrency and parallelism at all possible levels.

1.1 Types of Parallelism

The term "parallel processing" encompasses in its scope a wide variety of computer systems. One classification has been given by Flynn [Fly 66, Fly 72b], who divides computer systems into four categories:

(a) Single Instruction Single Data (SISD),

(b) Single Instruction Multiple Data (SIMD),

(c) Multiple Instruction Single Data (MISD), and

(d) Multiple Instruction Multiple Data (MIMD).

SISD covers the usual uniprocessor computers. Associative processors, processing ensembles, and array processors, such as the ILLIAC IV, fall in the SIMD category. Pipeline processors may be considered to be either of the SIMD, MISD, or MIMD architectures. Multiprocessors and multicomputer systems belong to the MIMD class.

1.2 Multiprocessors

Multiprocessors, the subject of this thesis, as distinct from multiple-computer systems, are not easy to define. The difference between a multiple-computer system and a multiprocessor is in the extent and degree of sharing : whereas the former consists of several separate and discrete computers, the latter is a single computer with multiple processing units.

Enslow [Ens 74, Ens 77] defines a multiprocessor as a system with the following characteristics:

(a) It contains two or more processing units of approximately comparable capabilities;

(b) All processors share access to a common memory (although some private memory may be allowed);

(c) All processors share access to input/output channels, control units, and devices;

(d) There is a single integrated operating system in overall control of all hardware and software; and

(e) There must be intimate interaction possible at both hardware and software operating levels.

Figure 1 depicts the basic structure of a multiprocessor system. Thus a multiprocessor has capabilities for the sharing of memory and input/output devices by all processors; the input/output devices also have complete access to memory. Hence the interconnection system has to support three types of communication : processor-memory, processor-I/O, and memory-I/O.

Although multiprocessors have all the three advantages of reliability, flexibility, and higher performance mentioned earlier, they pose a number of problems not encountered in single-processor systems. A multiprocessor system must have special facilities, both hardware and software, to resolve contention for shared resources. The operating system is larger and more complex than for uniprocessors. To properly exploit the available parallelism, tasks need to be divided into subtasks which can be executed in parallel. This makes scheduling considerably more complicated.

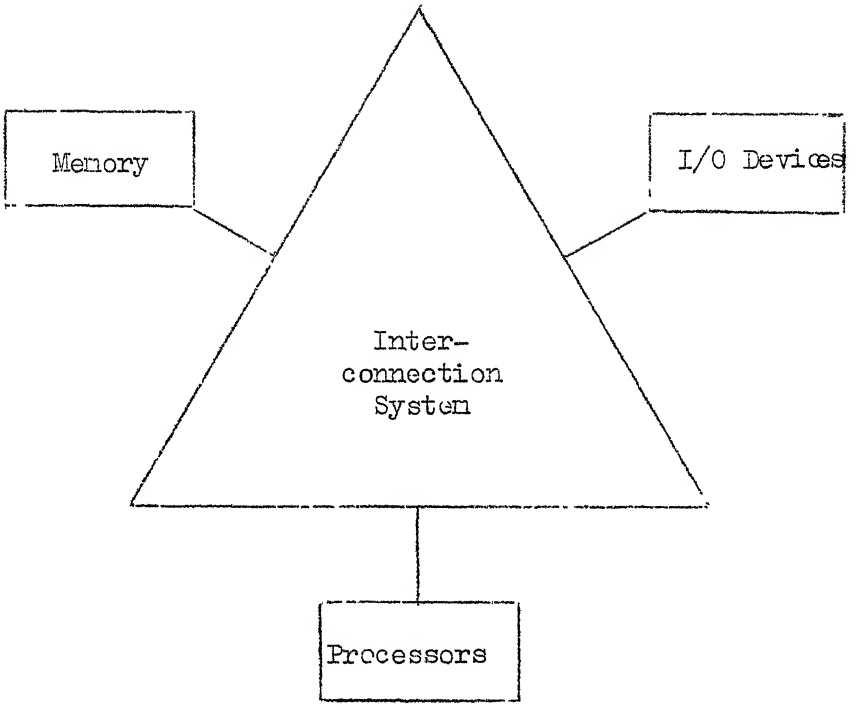


FIGURE 1: Structure of a Multiprocessor System.

At the hardware level, proper mechanisms have to be provided for communication between the various functional units. The interconnection system must have high bandwidth and must be reliable. A processor should have the capability of interrupting other processors. Efficient failure-detection is important for high reliability and both manual and automatic reconfiguration should be possible.

Performance monitoring and evaluation of multiprocessors are not only more complex than for uniprocessors, they are also more important. For a long time there had been an impression that multiprocessors were not capable of high performance and that they were not cost-effective. With better evaluation techniques, these mistaken notions are now being dispelled. Work on performance evaluation, however, still lags behind the advances in multiprocessor technology. The performance of a multiprocessor system is crucially dependent on the interconnection mechanism used for communication between the functional units. Hence the prime effort of ongoing research in this area is to devise better and more efficient interconnection schemes. The system designer must then have adequate tools available to enable him to evaluate and compare the performances of multiprocessors which use these various schemes. It is the problem of devising such evaluation techniques that we address ourselves to in this thesis.

1.3 Overview of the Thesis

We start in Chapter 2 by describing some of the important interconnection schemes used in multiprocessors, namely, the crossbar switch, time-shared bus, multiport memory/multibus, and a hybrid interconnection scheme used in the Pluribus multiprocessor built by Bolt, Beranek, and

Newman, Inc. (BBN), Cambridge, Mass., for the ARPA Network. Salient characteristics as well as the advantages and disadvantages of these schemes are discussed.

We begin Chapter 3 with a summary of existing work on analytic models for crossbar switch multiprocessors. Most of the past research on this topic has assumed that the memory references of each processor are uniformly distributed among all the memory modules. Although this assumption considerably simplifies the analysis, it is not very realistic, since programs generally exhibit the property of locality of references.

In Chapter 3, we develop a model for crossbar switch multiprocessors with local referencing, which reflects more closely the behavior of real systems. This model is analysed using both discrete and continuous Markov chain techniques, and expressions are derived for the multiprocessor performance. New expressions are also obtained for the performance in the traditional uniform reference model and are compared with other expressions available in the literature. Results of a simulation study are given to show the accuracy of the expressions for both models.

Almost all the work to date on computer modelling to analyse the performance of multiprocessor systems has been limited to the study of systems using a crossbar switch as the interconnection medium. As mentioned in the previous section, the tools of analytic modelling need to be improved to keep pace with the innovative development of new interconnection schemes. Keeping this aim in view, one of the main contributions of this thesis is the construction of analytic models for multiprocessors using the time-shared bus and the hybrid Pluribus scheme as the interconnection structures.

Chapter 4 describes a discrete Markov chain model for time-shared bus multiprocessors. An example is given to explain the detailed analysis technique and simulation results are presented to verify the results of the analysis.

A model for evaluating the performance of the Pluribus multiprocessor is described in Chapter 5. The Pluribus is a hybrid system containing a crossbar switch and a number of time-shared buses. The analytic model described there decomposes the system into its crossbar switch and time-shared bus components, simultaneously taking into account the complex interaction between these components. The crossbar switch is then analysed in terms of an existing model while the model of Chapter 4 is used to analyse the time-shared bus component. These results are synthesized to give the performance of the whole system. Graphical results are presented to show the effect of the various parameters of the system on its performance. Simulation results are given to verify the validity of the model.

Chapter 6 presents the conclusions and suggests the directions that future work in this area may take.

-

CHAPTER 2

MULTIPROCESSOR INTERCONNECTION STRUCTURES

Of paramount importance in a multiprocessor system is the communication mechanism and the mode of interconnection between its functional units, namely, the processors, memory modules, and the input/output units. Sharing of memory modules between multiple processors and I/O units results in conflicts between units desiring to access the same memory module at the same time. This phenomenon is called memory interference and it is the primary cause of degradation in the multiprocessor performance. Thus the main task of an analytic model for a multiprocessor is to estimate the amount of memory interference in the system.

In the models considered in this thesis, the effect of input/output units will not be modelled explicitly. This is because, in most cases, their effect on the overall performance of the system is insignificant [Str 70]. For example, transferring with four drums or 15 fixed head disks at full rate is comparable to the activity of one processor [Bel 71].

Some of the important interconnection media used in multiprocessors are the crossbar switch, time-shared bus, multiport memory/multibus, and a hybrid interconnection scheme used in the BBN Pluribus multiprocessor. There are a number of good papers which focus on multiprocessor interconnections [And 75, Bae 76, Ens, 74, Ens 77, Fer 73, Sea 75, Swa 76]. In this chapter, we shall discuss the salient features as well as some of the advantages and disadvantages of these interconnection schemes. This discussion, however, constitutes only one way of interpreting these various schemes. There are other ways of viewing these structures as is

obvious from the references cited above. Swan et al [Swa 76], for instance, regard these interconnection structures as mere variants of one fundamental structure, namely, the crossbar switch.

2.1 Crossbar Switch

In the crossbar switch organization, shown in Figure 2, any memory module can be connected to any processor. A full-time connection is established between the two units for the complete duration of the transfer. In the absence of a conflict, multiple connections are possible at a time. Such an organization is characterized by high bandwidth; in fact, among all the interconnection schemes, the crossbar switch has the potential for the highest total system transfer rate.

Since all the circuitry for conflict resolution is incorporated in the switch itself, the control logic of the memory modules is very simple. If the switch is distributed, the system can be made modular as well as reliable, and additional processors and/or memory modules can be added without too much difficulty.

However, the crossbar switch is extremely complicated and costly. It has been estimated that the cost of a switch for a large system is comparable to the cost of a few processors [Ens 74].

An important example of a multiprocessor system using a crossbar switch is C.mmp, the multimini processor built at Carnegie-Mellon University [Bel 71, Wul 72]. The TDC-316 multiprocessor under construction at the Tata Institute of Fundamental Research, Bombay [Jos 76, Nay 76], also employs a crossbar switch.

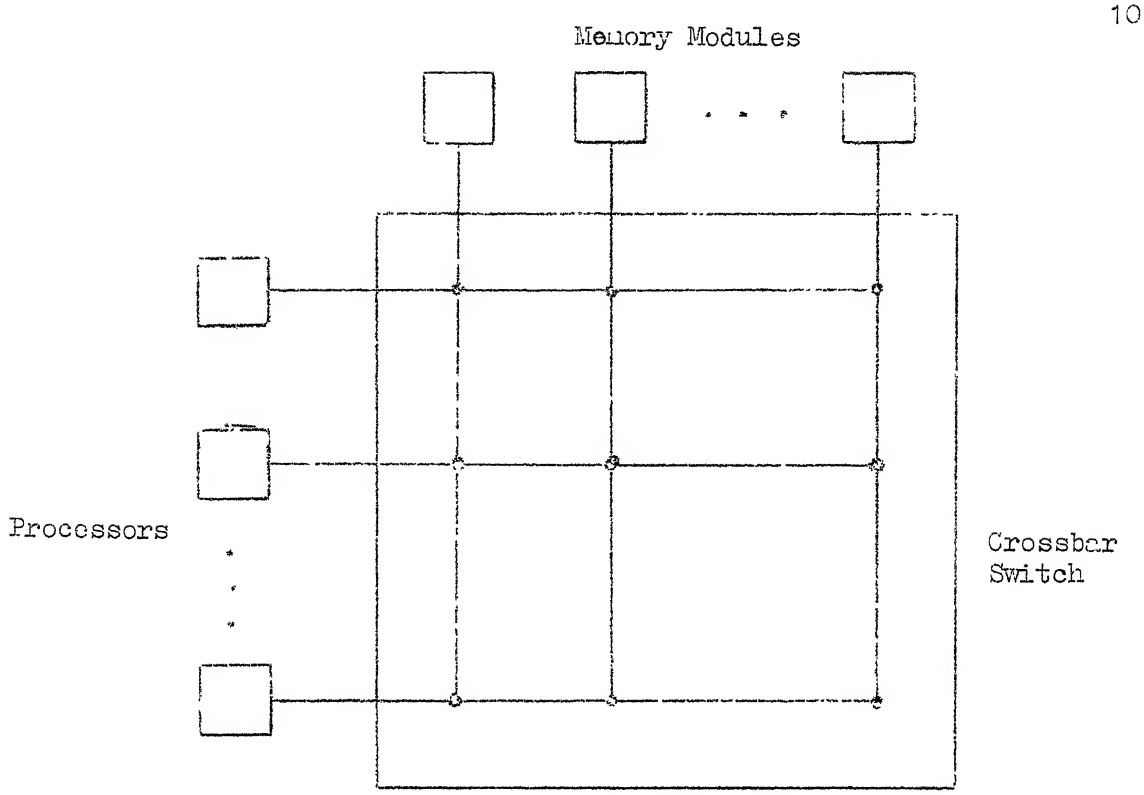


FIGURE 2: Crossbar Switch Configuration.

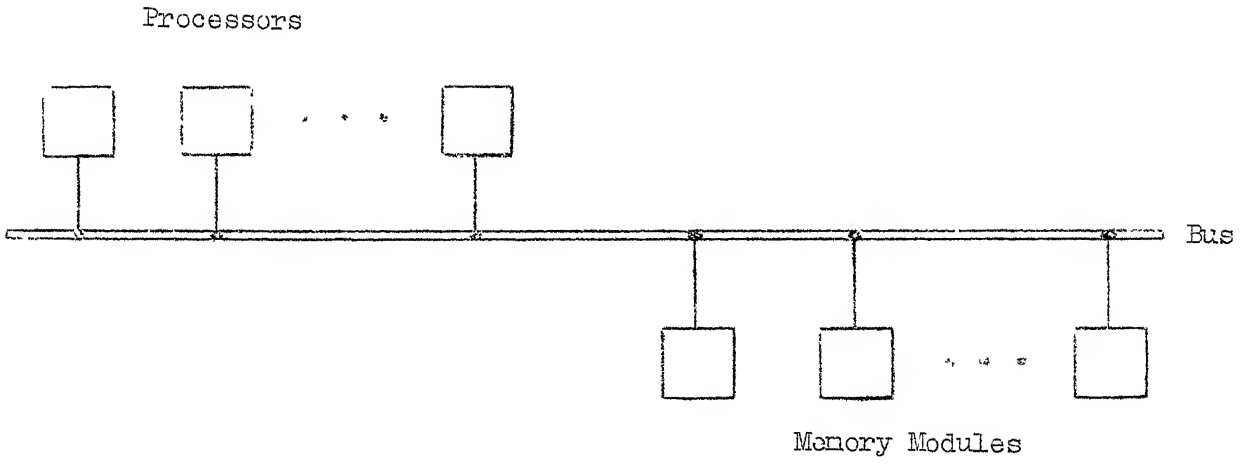


FIGURE 3: Time-Shared Bus Configuration.

2.2 Time-shared Bus

The time-shared bus system, shown in Figure 3, is one of the simplest and cheapest interconnection schemes. Here there are no continuous connections between functional units. All the units are connected in parallel to the bus which allows communication between one pair of units at a time. Since there is only one transfer path for all transfers, the system bandwidth and efficiency are low and the reliability is poor. It is very easy to physically modify the system configuration by adding or removing functional units. However, system expansion results in considerable degradation of the overall system performance.

For these reasons, the use of the time-shared bus is limited and it is not generally used for high-performance multiprocessors. Examples of systems using a time-shared bus are the PDP-11 and the Lockheed SUE. A good description of the operation of a bus may be found in [Thu 72] and in [Jan 71].

2.3 Multiport Memory/Multibus

The multiport memory/multibus system, shown in Figure 4, tries to overcome some of the disadvantages of the single time-shared bus. It is also less costly than the crossbar switch. For achieving high bandwidth, each processor may be assigned a dedicated bus, although this is poor from the point of view of reliability. If a bus bottleneck is present, expanding the number of buses increases the system throughput.

However, it is essential, in this organization, for the memory modules to have a number of access ports which makes the control logic of the memories more complex. The cabling and connector costs are large

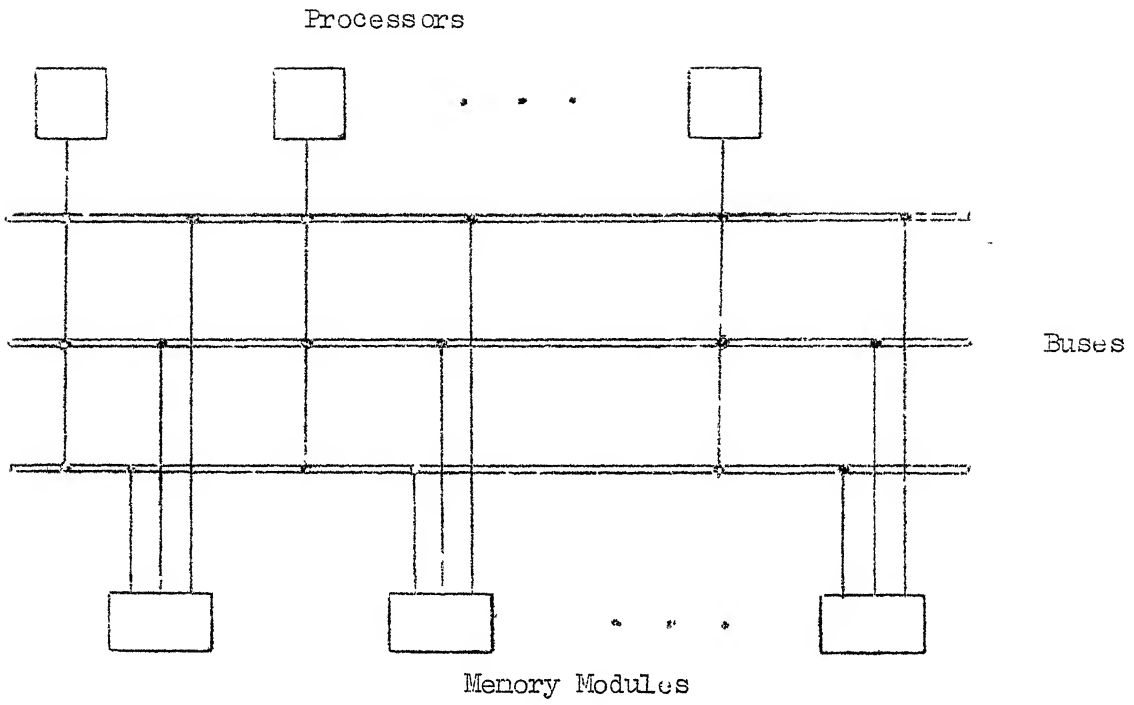


FIGURE 4: Multiport Memory/Multibus Configuration.

and become more pronounced as the system expands. The maximum configuration possible is also limited by the number of ports available on a memory module.

The UNIVAC 1108 and the IBM System 360 Model 67 are examples of such systems.

2.4 Pluribus Organization

This interesting unconventional scheme has been used by BBN for their Pluribus multiprocessor to be used as an Interface Message Processor for the ARPA Network [Bar 75, Hea 73, Orn 75]. This is a hybrid between the crossbar switch and the time-shared bus systems and is shown in Figure 5.

In this system, there are a number of processor buses, each containing a few processors and a few local memories connected by a time-shared bus. The local memories on a processor bus are accessible only to the processors on that bus. There are also additional memory buses which contain only memory modules. These memories are global and are accessible to any processor via the crossbar switch. In the system built by BBN, there are 7 processor buses each having 2 processors and 2 local memories, and 2 memory buses each having 2 global memories. The crossbar switch is distributed and is implemented by interconnecting units called bus couplers on various buses.

The Pluribus system is cheaper than the pure crossbar switch scheme and does not have the bandwidth limitations of the single bus scheme. However, for maximum efficiency, the bandwidths of the various components of the system must be carefully matched.

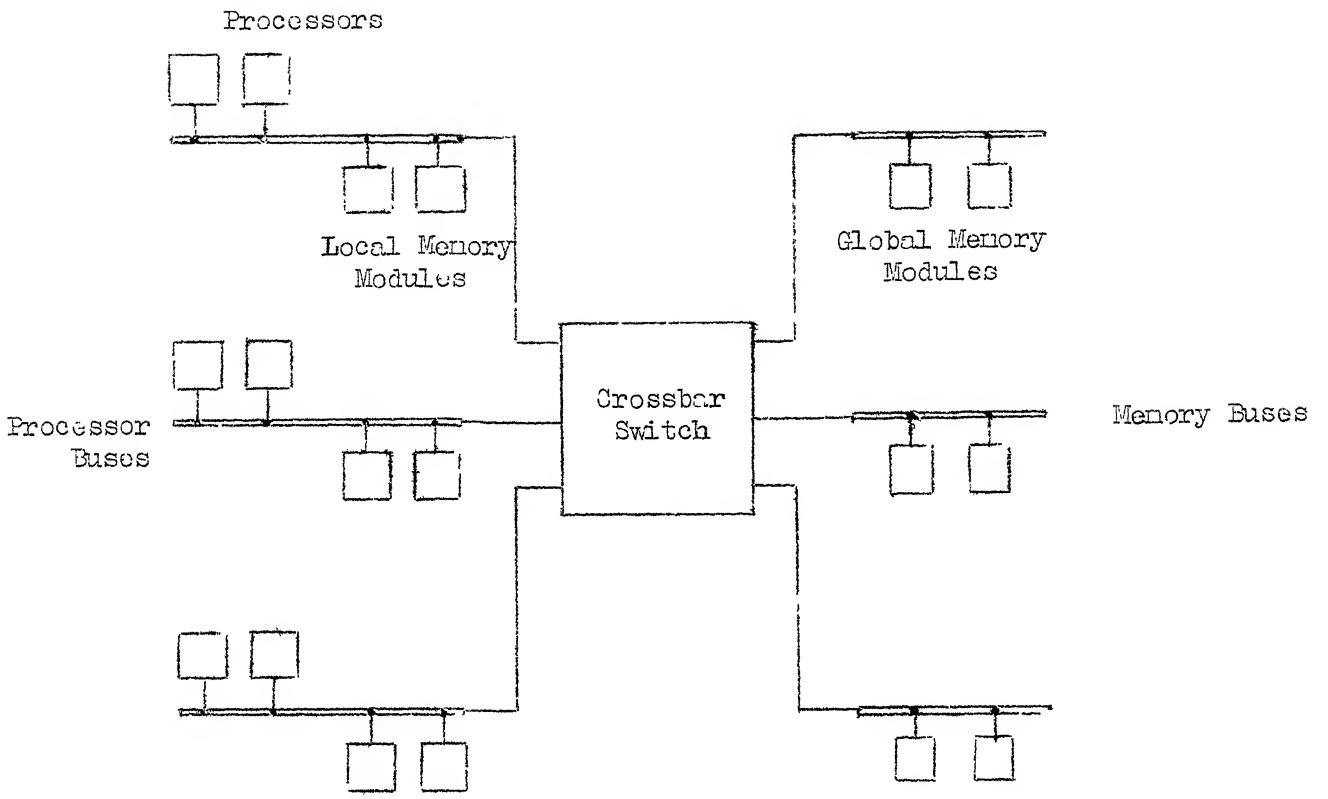


FIGURE 5 : PLURIBUS Configuration.

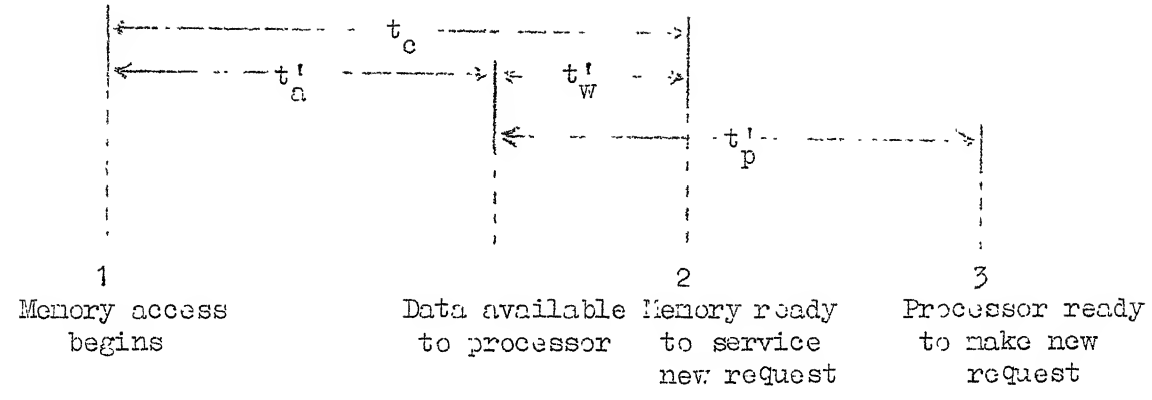
2.5 Definitions and Notations

A mathematical model of a system may be constructed at various levels of abstraction [Bha 76]. The interconnection mechanism in a multiprocessor forms the interface between the processors and the memory modules. This interface has a significant effect on the multiprocessor performance; for this reason, the models considered in this thesis are all at the level of the processor-memory interface.

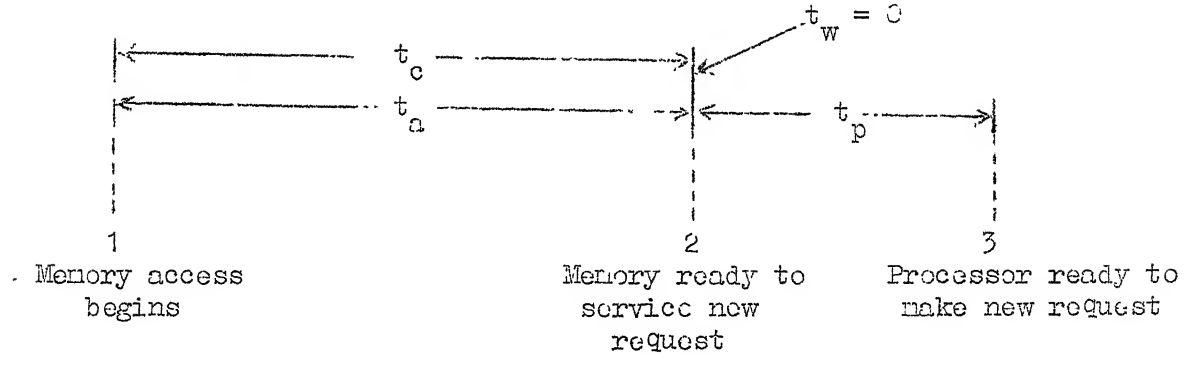
At this level, the interaction between the processors and memories must be very precisely defined. In general, processor behavior varies for different instructions. However, we shall not explicitly model these differences in instructions. We shall also make no distinction between the processing needed to decode an instruction and the processing corresponding to its execution. Instead we shall use the concept of a unit instruction, first proposed by Strecker [Str 70], which simply models the fetching of a word from memory followed by the processing of the word by a processor.

A diagrammatic representation of a unit instruction is shown in Figure 6(a). In this figure, t_p' represents the processing time of the processor, t_c is the memory cycle time, t_a' the memory access time and t_w' the memory rewrite time.

In most cases of interest to us, t_p' will generally be greater than or equal to t_w'. In these cases, to facilitate our discussion, a transformation may be made on the diagram of Figure 6(a) to give Figure 6(b). The memory now has an access time of t_c and zero rewrite time; the new processing time is $t_p = t_p' - t_w'$. This transformation introduces no change



(a)



(b)

FIGURE 6: Diagrammatic representation of a Unit instruction.

in the sense that the performance of the system in either case is the same. In both cases, the memory access begins at point 1, the memory is ready to service a new request at point 2, and the processor execution is completed at point 3. (This transformation was first used by Strecker [Str 70]). Thus, in all our models, without loss of generality, we shall assume the memory access time to be equal to the cycle time, and the rewrite time to be zero.

The performance measure used most often in this thesis is the Unit Instruction Execution Rate (UER) which is the total number of unit instructions executed by the system in unit time (generally 1 μsec.). We would like to emphasize, however, that other performance measures, such as utilization factor, percentage idle time, etc., can all be reduced to UER, and given one, all the others can be easily calculated, if necessary. Thus, processor utilization = t_p' x UER, memory utilization = t_c x UER etc.

-

CHAPTER 3

MODELS FOR CROSSBAR SWITCH SYSTEMS

In this chapter, we shall be concerned with analytic models for multiprocessors using a crossbar switch. In Section 3.1, we give a summary of past work in this area. All these existing models make the assumption that the memory references of each processor are uniformly distributed among the memory modules. Although this assumption considerably simplifies the analysis, it is not very realistic, since programs generally exhibit the property of locality of references. We shall propose a model with local referencing, which reflects more closely the behavior of real systems. Section 3.2 lists in detail the assumptions made for this model. In Section 3.3, we use discrete Markov chain techniques to analyse this model. Section 3.4 presents an alternative analysis using continuous Markov chain processes. Simulation results are given in Section 3.5. Finally, new expressions for the uniform reference model are developed in Section 3.6.

3.1 Review of Existing Models

In this section, we consider systems with p processors and m memory modules. We shall denote by t_p the average processing time of all processors (which are assumed to be identical); all memory modules are assumed to have equal constant cycle times t_c with access time t_a and rewrite time t_w.

Skinner and Asher [Ski 69] were the first to use Markov chain models to analyse multiprocessors. However, their study was limited to a small number of processors and memory modules, and they found it difficult to generalize their expressions for larger systems.

Strecker [Str 70] using some simplifying assumptions was able to give general approximate expressions. He considered three cases: (a) $t_p < t_w$, (b) $t_p = t_w$, and (c) $t_p > t_w$. Of these, the third case is important because many real systems fall in this category. Strecker gives the following expression for the performance of a multiprocessor system for case (c) ($t_p > t_w$):

$$UER = (m/t_c)(1 - (1 - P_m/m)^p)$$

$$\text{where } P_m + (m/p)\left(\frac{t_p - t_w}{t_c}\right)(1 - (1 - P_m/m)^p) - 1 = 0 \qquad (3.1)$$

In Chapter 5 we shall have occasion to use this equation. There we shall assume t_w to be zero; since t_p is always positive, our system will fall under case (c) and Equation (3.1) will be applicable to it. It should, however, be remembered that Strecker's analysis is approximate, not exact.

Bhandarkar [Bha 75] used discrete Markov chain models to analyse cases (b) and (c) above. He used this analysis to write a program to calculate exact values for the system performance. However, his program is time-consuming for even moderately-sized systems. Bhandarkar simplified Strecker's expression for case (b) in the light of the results available from his program. Bhandarkar and Fuller analysed multiprocessors using a continuous-time Markov chain

model.

In this chapter we shall consider only those multiprocessor systems in which the memory is partitioned into modules by the higher order bits of the address. Memories interleaved by the low order address bits have been studied by Burnett and Coffman [Bur 70, Bur 73, Bur 75], also jointly with Snowdon [Cof 71], and by Sastry and Kain [Sas 75]. It should be noted that if we assume uniformly distributed memory references by each processor, then the behavior of low-order interleaved memories is no different from that of high-order interleaved memories. Baskett and Smith [Bas 76] have given asymptotic results for low-order interleaved memories with uniformly distributed references. Thus their physical model is the same as that of Bhandarkar, with the difference that they have studied its asymptotic behavior.

3.2 Assumptions for Local Reference Model

The following major assumptions characterize the local reference model developed in this chapter:

Assumption 1: The system has p processors and m memory modules. All processors and all memory modules are identical. This will be referred to as a p x m system.

Assumption 2: Instructions of the processors are modelled using the concept of the unit instruction defined in Section 2.5.

Assumption 3: All memory modules have equal constant cycle times and their operation is synchronized with no overlapping of read/write cycles. The access time of each module is equal to its cycle time, and the rewrite time is zero (see Figure 6(b)). The processing time of each

processor is assumed to be zero.

Assumption 4: The processors and memories are connected by a crossbar switch which permits every processor to have access to every memory module. All memory modules are simultaneously accessible so that, under no conflict, a maximum of min(p,m) words can be fetched simultaneously. The switch is assumed to have zero delay. However, crossbar switches with nonzero delay may be modelled by simply adding the delay to t_c, the memory cycle time.

Assumption 5: From each memory module only one word can be fetched at a time. If two or more processors simultaneously make requests for the same memory module, only one of these requests can be served in the next memory cycle. The other processors are queued up at the module to be served in subsequent cycles.

Assumption 6: Consecutive addresses in memory are mapped into the same module modulo the module size. Thus the high-order bits of an address determine the module to which the address belongs.

Assumption 7: Successive requests of a processor follow the pattern described below. If the k-th request of a processor is for memory module i, then its (k + 1)st request will be for module i with probability α, and for module j($j \neq i$) with probability $(1 - \alpha)/(m - 1)$. Thus all memory modules except module i are accessed with equal probability. Probability α is a constant and is equal for all processors.

It should be noted that if $\alpha = 1/m$ then all memory modules are accessed with equal probability, and this model reduces to the uniform reference model analysed by Bhandarkar [Bha 75]. However, in general, α will not equal $1/m$, in which case we shall call this the local reference model.

If α is large compared to $1/m$, the processor will tend to access the same memory module repeatedly until it changes to a different module, and the same behavior is repeated. It is our belief that this model is more representative of real-life multiprocessor systems than the uniform reference model. A multiprocessor system generally works in a multiprogramming environment in which each processor executes a more-or-less independent task. Thus each processor would concentrate its attention on blocks of consecutive addresses which, in our model, are mapped into the same module. Thus the probability of consecutive references being to the same module is quite high. Occasionally a task may be split into one or more modules; references may also be made to the executive which may reside in a different module. But this happens relatively infrequently; programs are also mostly sequential in nature and present-day programming styles emphasize modular programs. Hence the parameter α, though not equal to 1, will be quite close to it. It seems reasonable to assume that most such environments will have $\alpha > 0.75$. We shall show later that the multiprocessor performance is more or less unaffected by the value of α so long as α lies in this range. However, the performance of systems with $\alpha > 0.75$ is worse than that predicted by the uniform reference model.

The performance measure used in this chapter is the Average Number of Busy Memory Modules (ANBM). This is the average number of memory modules that are busy during a memory cycle. Using this measure will permit us to conveniently compare our results with other results available in the literature. Its relation with the Unit Instruction Execution Rate (UER) is given by

$$UER = ANBM/t_c .$$

3.3 Discrete Markov Chain Analysis

In this section we shall analyse the local reference model, defined by the assumptions of the previous section, using discrete Markov chain techniques. An excellent description of Markov processes may be found in Kleinrock [Kle 75]. Bhandarkar [Bha 75] has developed a systematic approach to the use of the discrete Markov chain technique for analysing memory interference in multiprocessor systems. We shall use this technique in the analysis of this section, and again in Chapter 4 for analysing the model for the time-shared bus.

The exact analysis of the Markov chain model is very complex, even for the uniform reference model. For this reason, Bhandarkar did not attempt to derive general expressions for the system performance with p and n as parameters. Instead, he wrote a program to compute the Average Number of Busy Memory Modules for any given p x n system.

In this section, we shall derive such expressions for the local reference model with n as a parameter for small, constant values of p (such as 2 or 3), and correspondingly, with p as a parameter for small, constant values of n. Approximations of these expressions will then be

generalized to hold for all values of p and n. We shall use the same approach with the uniform reference model in Section 3.6. It should be noted that the local reference model has α as an additional parameter making the analysis more complicated. Our interest will, however, lie in systems for which α exceeds 0.75.

At any given time, the state of a p x n system can be characterized by the lengths of the queues at each memory module. Following Bhandarkar's notation, this state is denoted by an n-tuple $(k_1, k_2, \ldots, k_n)$, where

$$\sum_{i=1}^{n} k_i = p,$$

and $0 \leq k_i \leq p$ for $1 \leq i \leq n$. Integer k_i represents the number of processors waiting in the queue at module i (including the processor being served). Since all processors are identical, a number of these states are equivalent, such as states (2,2,1), (2,1,2) and (1,2,2). Every such equivalence class will be called a reduced state. In the notation of a reduced state, we shall generally omit all 0's, e.g., state (2,1,0,0) will be written simply as (2,1). For any given value of n, this notation is unambiguous.

Let us consider a 2 x n system, in which there are two processors and $n \geq 2$ memory modules. This system has only two reduced states, $s_1 = (2)$ and $s_2 = (1,1)$. Consider state $s_1 = (2)$. At the end of a memory cycle, the resultant partial state is (1) with one free processor to be reassigned. This may be assigned to the same memory module with probability α and to a different module with probability $(1 - \alpha)$. Thus

transitions from state s_1 to states s_1 and s_2 will occur with probabilities α and $(1 - \alpha)$ respectively. Following a similar procedure for state s_2, the transition matrix T can be shown to be:

$$T = \begin{bmatrix} \alpha & 1 - \alpha \\ \dfrac{(1 - \alpha)(n\alpha + n - 2)}{(n - 1)^2} & \dfrac{n^2 + n(\alpha^2 - 3) + 3 - 2\alpha}{(n - 1)^2} \end{bmatrix}$$

By computing the steady-state probabilities for this Markov chain, it can be shown that the Average Number of Busy Memory Modules for a 2 x n system is given by

$$ANBM = \frac{n(2n + \alpha - 3)}{n(n + \alpha - 1) - 1} \tag{3.2}$$

Let us now study a p x 2 system having two memory modules and $p \geq 2$ processors. This system has $\lfloor p/2 \rfloor + 1$ states[1]:

$$(p),\ (p - 1,\ 1),\ (p - 2, 2),\ \ldots,\ (\lfloor (p+1)/2 \rfloor,\ \lfloor p/2 \rfloor).$$

For example, if $p = 8$, then the states are (8), (7,1), (6,2), (5,3), and (4,4); if $p = 9$, then the states are (9), (8,1), (7,2), (6,3), and (5,4).

The transition matrix can now be obtained, and it can be shown that for a p x 2 system:

$$ANBM = \frac{2(p + \alpha - 1)}{p + 2\alpha - 1} \tag{3.3}$$

If we substitute $\alpha = 1$ in equations (3.2) and (3.3), we get respectively:

$$ANBM = 2 - \frac{2}{n+1} \tag{3.4}$$

and

$$ANBM = 2 - \frac{2}{p+1} \tag{3.5}$$

Footnote:

[1] $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x.

We shall show in Section 3.5 that when α lies in the range 0.75 to 0.95, Equations (3.2) and (3.3) can be approximated, without any significant loss in accuracy, with Equations (3.4) and (3.5) respectively.

Now consider a 3 x n system with three processors and $n \geq 3$ memory modules. This system becomes exceedingly difficult to analyse for an arbitrary value of probability α using the method employed in the analysis of the 2 x n and p x 2 systems, because of the large number of states involved. However, if we assume $\alpha = 1$, the problem is more tractable, and it can be shown that for a 3 x n system with $\alpha = 1$

$$\text{ANBM} = 3 - \frac{6}{n+2} \tag{3.6}$$

That this is a good approximation to the actual value when α lies in the range 0.75 to 0.95 is borne out by the simulation results discussed in Section 3.5.

A general expression suggested by the three Equations (3.4), (3.5), and (3.6) is that in a p x n system with $\alpha = 1$ the Average Number of Busy Memory Modules should be given by

$$\text{ANBM} = p - \frac{p(p-1)}{n+p-1} = \frac{np}{n+p-1} \tag{3.7}$$

It should be noted that this equation is symmetric with respect to n and p. The nature of the actual values of ANBM and the accuracy of these approximations will be explored in Section 3.5.

3.4 Continuous Markov Chain Analysis

In this section, we shall use a continuous-time Markov chain model to analyse the local reference model. Interestingly, the exact solution for this method is the same as Equation (3.7) which

was an approximate solution for the discrete method.

Continuous-time Markov chains were used by Bhandarkar and Fuller [Bha 73] to analyse the uniform reference model. To use this technique, we need to abandon the assumption of constant memory cycle time and assume that memory cycle times are exponentially distributed. Although this assumption is not very realistic, it may be useful to view the resulting expression as a lower bound on the system performance [Bha 73]. Moreover, this method gives us an expression that is valid for all values of α.

The formulae used here were derived by Jackson [Jac 63], and Gordon and Newell [Gor 67]; we shall, however, use the notation of Kleinrock [Kle 76, Section 4.12]. Our model is now viewed as a closed queuing network with n service centers and p permanent customers. Transitions from one center to another are determined by a routing probability matrix R. The element r_{ij} of this matrix gives the probability of going to center j on completion of service at center i and, in our model, is equal to α when $i = j$ and $(1 - \alpha)/(n - 1)$ when $i \neq j$. States are denoted by a vector $\vec{k} = (k_1, k_2, \ldots, k_n)$ as in the discrete model. The equilibrium probability $p(k_1, k_2, \ldots, k_n)$ is given by

$$p(k_1, k_2, \ldots, k_n) = \frac{1}{G(p)} \prod_{i=1}^{n} x_i^{k_i}$$

where

$$G(p) = \sum_{\vec{k} \in A} \prod_{i=1}^{n} x_i^{k_i},$$

A is that set of vectors $\vec{k}$ for which

$$\sum_{i=1}^{n} k_i = p,$$

$x_i = \lambda_i/\mu_i$, λ_i are the solutions of $\lambda_i = \sum_{j=1}^{n} \lambda_j R_{ji}$, and μ_i are the mean service rates of the service centers.

Substituting for R_{ji} in the last equation, we get

$$\lambda_i = \alpha \lambda_i + \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{(1-\alpha)}{(n-1)} \cdot \lambda_j$$

or

$$\lambda_i = \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} \lambda_j$$

which gives

$$\lambda_i = \frac{1}{n} \sum_{j=1}^{N} \lambda_j .$$

Thus all λ_i's are equal and independent of α. From here on, the analysis is exactly the same as done by Bhandarkar and Fuller [Bha 73]. Solving for the equilibrium probabilities, we find that

$$p(k_1, k_2, \ldots, k_n) = 1/\binom{n+p-1}{n-1}$$

for all $\vec{k}$. Thus, all the states of the system are equally likely. This gives the Average Number of Busy Memories as

$$ANBM = \frac{np}{n+p-1} .$$

This equation is identical to Equation (3.7). It was first derived in [Bha 73] for the uniform reference model, i.e., when $\alpha = 1/n$, which is a particular case of our derivation.

We thus find that under the assumption of exponentially distributed memory cycle times, the performance of the system is independent of σ. Equation (3.7) may be viewed as a lower bound on the performance of real-life systems in which the cycle time is not exponentially distributed. Similarly, the discrete Markov chain model gives an upper bound since the processing time of a real-life system is never a constant and is better approximated by the exponential distribution [Bha 75].

3.5 Simulation Results

Simulation studies were conducted to validate the local reference model and to provide the basis for comparing the expressions derived in the earlier sections of this chapter. The programs were written in FORTRAN IV and run on an IBM 7044. To find the steady-state system performance, the number of busy memory modules in a cycle was averaged over a total of 5000 memory cycles. This amounted to the processing of between 7000 and 33000 instructions (approximately) by the multiprocessor system, depending on the number of processors and memory modules.

Figure 7 shows the Average Number of Busy Memory Modules plotted as a function of α for various values of p and m. This figure clearly demonstrates that for a given multiprocessor system, ANBM falls as α increases from 0 to 1. However, over the range $\alpha > 0.75$, the variation in ANBM is very small. Thus the system performance may be accurately represented by the average value of the ANBM figures in this range.

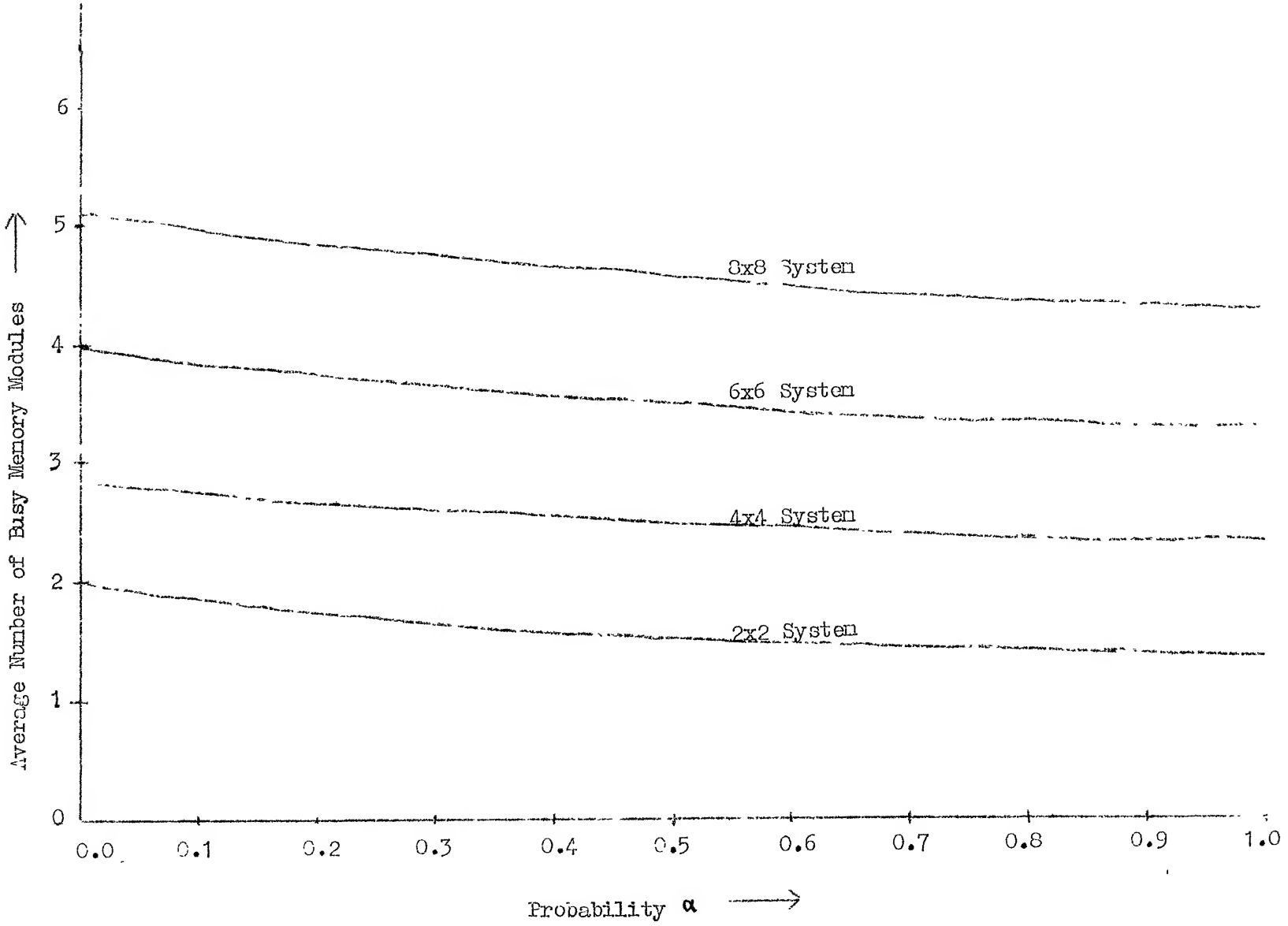


FIGURE 7: Effect of program locality.

As mentioned in Section 3.2, a multiprocessor system is generally expected to have $\alpha > 0.75$. The averages of the values of AMBM corresponding to $\alpha = 0.75$, 0.8, 0.85, 0.9, and 0.95 were computed and these are shown in Table I for various p x m systems. These averages are compared with values obtained from Equation (3.7). In no case does the error exceed 4 percent. Thus, Equation (3.7) provides a very good approximation for the performance of real-life systems. The same is true of Equations (3.4), (3.5), and (3.6), since they are particular cases of (3.7).

A comparison of the performances predicted by the uniform reference and local reference models is shown in Figure 8. The values used for the discrete Markov chain uniform reference model are taken from [Bha 75], while those for the local reference model are computed from Equation (3.7). As we saw in the previous section, Equation (3.7) forms a lower bound on the performance for all values of α. It also forms an approximate upper bound for systems with $\alpha > 0.75$. Thus, for these systems, this equation is a very good estimate of the performance. The upper bound for the uniform reference model is higher than Equation (3.7); therefore the performance predicted by this model is generally more optimistic than it would be for real-life systems (with $\alpha > 0.75$). Simulation results for the case $\alpha = 0$ are also shown in Figure 8. It is evident from the figure that the performance of multiprocessor systems would be improved if programs have addressing patterns that would make α close to 0.

TABLE I

AVERAGE NUMBER OF BUSY MEMORY MODULES (ANBM)
FOR THE LOCAL REFERENCE MODEL

Number of Processors p = 2, 3, ..., 8 (rows)
Number of Memory Modules n = 2, 3, ..., 10 (columns)

(a) Simulation results averaged for α = 0.75, 0.8, 0.85, 0.9, 0.95

1.374	1.520	1.624	1.694	1.738	1.770	1.783	1.814	1.822
1.529	1.844	2.046	2.203	2.300	2.389	2.444	2.508	2.541
1.623	2.052	2.347	2.552	2.706	2.833	2.964	3.019	3.127
1.680	2.207	2.554	2.844	3.075	3.225	3.351	3.520	3.657
1.710	2.320	2.719	3.052	3.336	3.531	3.738	3.964	4.068
1.770	2.406	2.863	3.255	3.620	3.843	4.115	4.276	4.455
1.781	2.467	3.018	3.422	3.757	4.116	4.317	4.587	4.749

TABLE I (Continued)

(b) Approximate values calculated from Equation (3.7)

1.3333	1.5000	1.6000	1.6667	1.7143	1.7500	1.7778	1.8000	1.8182
1.5000	1.8000	2.0000	2.1429	2.2500	2.3333	2.4000	2.4545	2.5000
1.6000	2.0000	2.2857	2.5000	2.6667	2.8000	2.9091	3.0000	3.0769
1.6667	2.1429	2.5000	2.7778	3.0000	3.1818	3.3333	3.4615	3.5714
1.7143	2.2500	2.6667	3.0000	3.2727	3.5000	3.6923	3.8571	4.0000
1.7500	2.3333	2.8000	3.1818	3.5000	3.7692	4.0000	4.2000	4.3750
1.7778	2.4000	2.9091	3.3333	3.6923	4.0000	4.2667	4.5000	4.7059

(c) Percentage Error

2.9622	1.3158	1.4778	1.6116	1.3636	1.1299	0.2916	0.7718	0.2086
1.8967	2.3861	2.2483	2.7281	2.1739	2.3315	1.8003	2.1332	1.6135
1.4171	2.5341	2.6118	2.0376	1.4523	1.1648	1.8522	0.6293	1.6022
0.7917	2.9044	2.1143	2.3277	2.4390	1.3395	0.5282	1.6619	2.3407
0.2515	3.0172	1.9235	1.7038	1.8975	0.8779	1.2226	2.6968	1.6716
1.1299	3.0216	2.2005	2.2488	3.3149	1.9204	2.7947	1.7774	1.7957
0.1797	2.7158	3.6083	2.5921	1.7221	2.8183	1.1652	1.8967	0.9076

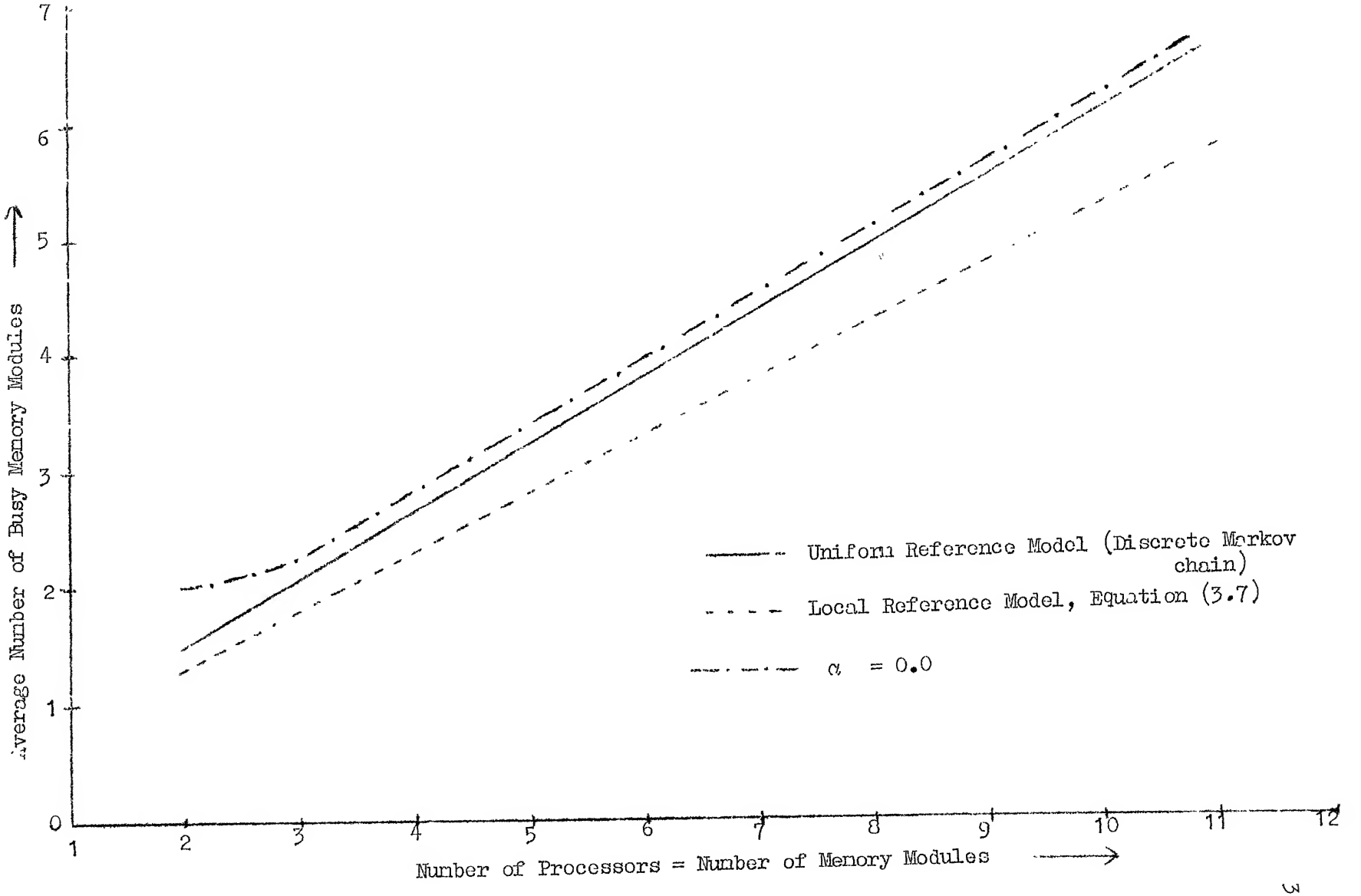


FIGURE 8: Comparison of uniform reference and local reference models

3.6 Uniform Reference Model

In this section, we shall derive expressions for the uniform reference model using the method followed in Section 3.3. Since the local reference model becomes equivalent to the uniform reference model upon substituting $\alpha = 1/n$, we get straightaway from Equations (3.2) and (3.3) that for uniform reference models

$$ANBM = 2 - 1/n \tag{3.8}$$

and

$$ANBM = 2 - 1/p \tag{3.9}$$

for the 2 x n and p x 2 systems respectively.

A similar discrete Markov chain analysis may be done for the 3 x n system ($n \geq 3$) to give

$$ANBM = 3 - \frac{3}{n} + \frac{1}{n^3-n^2+n} \tag{3-10}$$

in the case of uniform reference models. Note that all three expressions (3.8), (3.9), and (3.10) are exact. In (3.10), the last term becomes small when n increases, so we may write as an approximation:

$$ANBM = 3 - 3/n \tag{3-11}$$

Following a similar process, for a 4 x n system ($n \geq 4$) we find that[2]

$$ANBM = 4 - \frac{6}{n} + \frac{7n^3 - 12n^2 + 14n - 12}{n(n^5-3n^4+8n^3-11n^2+8n-4)} \tag{3.12}$$

When n is large, Equation (3.12) may be approximated by

$$ANBM = 4 - 6/n \tag{3.13}$$

Footnote:

[2] We owe the correct form of this equation to Dr. D.P. Bhandarkar.

A general expression, suggested by Equations (3.8), (3.11), and (3.13), is that in a p x n system ($n \geq p$) the Average Number of Busy Memory Modules (for a uniform reference model) should approximately be given by

$$ANBM = p - p(p-1)/2n \tag{3.14}$$

However, since we know from [Bha 75] that the performances of an A x B system and a B x A system are almost equal, we may write

$$ANBM = j - \frac{j(j-1)}{2i} \tag{3.15}$$

where $i = max(n,p)$ and $j = min(n,p)$.

Let us now compare expression (3.15) with two others available in the literature for the uniform reference model. First, Strecker's expression [Str 70], as modified by Bhandarkar [Bha 75], is:

$$ANBM = i\left[1 - (1 - \frac{1}{i})^j\right] \tag{3.16}$$

where $i = max(n,p)$ and $j = min(n,p)$. Second, an asymptotic expression given by Baskett and Smith [Bas 76] is:

$$ANBM = n + p - (n^2 + p^2)^{\frac{1}{2}} \tag{3.17}$$

In order to compare expressions (3.15), (3.16), and (3.17), we have used the exact numerical results given by Bhandarkar [Bha 75] and beyond Bhandarkar's with results obtained in the simulation study described in Section 3.5 (with 1/n substituted for α). Table II(a) gives the values of ANBM for p x n systems with $2 \leq p \leq 10$ and $2 \leq n \leq 12$. For $p \leq 8$ and $n \leq 8$ we have used the exact values of Bhandarkar. The rest of the entries were obtained by simulation. The values obtained from

TABLE II

AVERAGE NUMBER OF BUSY MEMORY MODULES (ANBM)
FOR THE UNIFORM REFERENCE MODEL

Number of Processors p = 2, 3, ..., 10 (rows)
Number of Memory Modules n = 2, 3, ..., 12 (columns)

(a) Bhandarkar's exact results (for $p \leq 8$ and $n \leq 8$)
and Simulation results (for $p > 8$ or $n > 8$)

1.5000	1.6667	1.7500	1.8000	1.8333	1.8571	1.8750	1.8889	1.9000	1.9091	1.9167
1.6667	2.0476	2.2692	2.4095	2.5054	2.5748	2.6272	2.674	2.697	2.710	2.756
1.7500	2.2701	2.6210	2.8630	3.0365	3.1657	3.2652	3.349	3.402	3.469	3.494
1.8000	2.4102	2.8633	3.1996	3.4530	3.6482	3.8019	3.930	4.034	4.092	4.176
1.8333	2.5059	3.0370	3.4533	3.7809	4.0415	4.2518	4.417	4.546	4.690	4.779
1.8571	2.5751	3.1663	3.6486	4.0418	4.3636	4.6292	4.859	5.041	5.185	5.306
1.8750	2.6274	3.2657	3.8024	4.2521	4.6294	4.9471	5.185	5.456	5.545	5.824
1.8889	2.663	3.331	3.881	4.393	4.864	5.191	5.550	5.814	6.042	6.268
1.9000	2.708	3.391	4.022	4.580	5.052	5.413	5.802	6.092	6.364	6.574

TABLE II (Continued)

(b) Bhandarkar's formula, Equation (3.16)

1.5000	1.6667	1.7500	1.8000	1.8333	1.8571	1.8750	1.8889	1.9000	1.9091	1.9167
1.6667	2.1111	2.3125	2.4400	2.5278	2.5918	2.6406	2.6790	2.7100	2.7355	2.7569
1.7500	2.3125	2.7344	2.9520	3.1065	3.2216	3.3105	3.3813	3.4390	3.4859	3.5272
1.8000	2.4400	2.9520	3.3616	3.5887	3.7613	3.8967	4.0056	4.0951	4.1699	4.2333
1.8333	2.5278	3.1065	3.5887	3.9906	4.2240	4.4096	4.5606	4.6856	4.7908	4.8805
1.8571	2.5918	3.2216	3.7613	4.2240	4.6206	4.8584	5.0538	5.2170	5.3553	5.4738
1.8750	2.6406	3.3105	3.8967	4.4096	4.8584	5.2511	5.4923	5.6953	5.8684	6.0176
1.8889	2.6790	3.3813	4.0056	4.5606	5.0538	5.4923	5.8820	6.1258	6.3349	6.5162
1.9000	2.7100	3.4390	4.0951	4.6856	5.2170	5.6953	6.1258	6.5132	6.7590	6.9732

(c) Percentage Error

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	3.1012	1.9082	1.2658	0.8941	0.6602	0.5100	0.1870	0.4820	0.9410	0.0327
0.0000	1.8678	4.3266	3.1086	2.3053	1.7658	1.3874	0.9645	1.0876	0.5160	0.9502
0.0000	1.2364	3.0978	5.0631	3.9299	3.1002	2.4935	1.9237	1.5146	1.9037	1.3721
0.0000	0.8739	2.2884	3.9209	5.5463	4.5157	3.7114	3.2511	3.0708	2.1493	2.1239
0.0000	0.6485	1.7465	3.0889	4.5079	5.8896	4.9512	4.0091	3.4914	3.2845	3.1625
0.0000	0.5024	1.3718	2.4800	3.7041	4.9466	6.1450	5.9267	4.3860	3.9575	3.3242
0.0000	0.6008	1.5101	3.2105	3.8152	3.9021	5.8043	5.9820	5.3629	4.8477	3.9598
0.0000	0.0739	1.4155	1.8175	2.3057	3.2660	5.2152	5.5808	6.9140	6.2068	6.0724

TABLE II (Continued)

(d) Baskett and Smith's formula, Equation (3.17)

1.1716	1.3944	1.5279	1.6148	1.6754	1.7199	1.7538	1.7805	1.8020	1.8197	1.8345
1.3944	1.7574	2.0000	2.1690	2.2918	2.3842	2.4560	2.5132	2.5597	2.5982	2.6307
1.5279	2.0000	2.3431	2.5969	2.7889	2.9377	3.0557	3.1511	3.2297	3.2953	3.3509
1.6148	2.1690	2.5969	2.9289	3.1898	3.3977	3.5660	3.7044	3.8197	3.9170	4.0000
1.6754	2.2918	2.7889	3.1898	3.5147	3.7805	4.0000	4.1833	4.3381	4.4700	4.5836
1.7199	2.3842	2.9377	3.3977	3.7805	4.1005	4.3699	4.5982	4.7934	4.9616	5.1076
1.7538	2.4560	3.0557	3.5660	4.0000	4.3699	4.6863	4.9584	5.1938	5.3985	5.5778
1.7805	2.5132	3.1511	3.7044	4.1833	4.5982	4.9584	5.2721	5.5464	5.7873	6.0000
1.8020	2.5597	3.2297	3.8197	4.3381	4.7934	5.1938	5.5464	5.8579	6.1339	6.3795

(e) Percentage Error

21.8933	16.3377	12.6914	10.2889	8.6129	7.3879	6.4640	5.7388	5.1579	4.6828	4.2886
16.3377	14.1727	11.8632	9.9813	8.5256	7.4025	6.5164	6.0135	5.0908	4.1255	4.5464
12.6914	11.8982	10.6028	9.2944	8.1541	7.2022	6.4161	5.9092	5.0647	5.0072	4.0956
10.2889	10.0075	9.3039	8.4604	7.6224	6.8664	6.2048	5.7405	5.3123	4.2766	4.2146
8.6129	8.5438	8.1692	7.6304	7.0407	6.4580	5.9222	5.2909	4.5753	4.6908	4.0887
7.3879	7.4133	7.2198	6.8766	6.4649	6.0294	5.6014	5.3674	4.9117	4.3086	3.7392
6.4640	6.5236	6.4305	6.2171	5.9288	5.6055	5.2718	4.3703	4.8057	4.3667	4.2273
5.7388	5.6252	5.4008	4.5504	4.7735	5.4646	4.4808	5.0072	4.6027	4.2155	4.2757

TABLE II (Continued)

(f) ANBM as given by Equation (3.15)

1.5000	1.6667	1.7500	1.8000	1.8333	1.8571	1.8750	1.8839	1.9000	1.9091	1.9167
1.6667	2.0000	2.2500	2.4000	2.5000	2.5714	2.6250	2.6667	2.7000	2.7273	2.7500
1.7500	2.2500	2.5000	2.8000	3.0000	3.1429	3.2500	3.3333	3.4000	3.4545	3.5000
1.8000	2.4000	2.8000	3.0000	3.3333	3.5714	3.7500	3.8889	4.0000	4.0909	4.1667
1.8333	2.5000	3.0000	3.3333	3.5000	3.8571	4.1250	4.3333	4.5000	4.6364	4.7500
1.8571	2.5714	3.1429	3.5714	3.8571	4.0000	4.3750	4.6667	4.9000	5.0909	5.2500
1.8750	2.6250	3.2500	3.7500	4.1250	4.3750	4.5000	4.8889	5.2000	5.4545	5.6667
1.8889	2.6667	3.3333	3.8889	4.3333	4.6667	4.8889	5.0000	5.4000	5.7273	6.0000
1.9000	2.7000	3.4000	4.0000	4.5000	4.9000	5.2000	5.4000	5.5000	5.9091	6.2500

(g) Percentage Error

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	2.3247	0.8461	0.3943	0.2155	0.1320	0.0837	0.2730	0.1112	0.6384	0.2177
0.0000	0.8854	4.6166	2.2005	1.2020	0.7202	0.4655	0.4688	0.0588	0.4180	0.1717
0.0000	0.4232	2.2107	6.2383	3.4666	2.1051	1.3651	1.0458	0.8428	0.0269	0.2227
0.0000	0.2354	1.2183	3.4749	7.4294	4.5627	2.9823	1.8950	1.0119	1.1429	0.6068
0.0000	0.1437	0.7390	2.1159	4.5697	8.3326	5.4912	3.9576	2.7971	1.8149	1.0554
0.0000	0.0913	0.4808	1.3781	2.9891	5.4953	9.0376	5.7107	4.6921	3.3747	2.7009
0.0000	0.1389	0.0690	0.2036	1.3590	4.0563	5.8197	9.9099	7.1207	5.2085	4.2757
		0.2654	0.5470	1.7467	3.0087	3.9350	6.9286	9.7177	7.1480	4.9285

Equations (3.16), (3.17), and (3.15) and their comparison with the exact results are shown in Table II.

It can be seen that Baskett and Smith's expression (3.17) is highly inaccurate for small values of m and p. Its accuracy improves as m and p increase. Both Bhandarkar's expression (3.16) and our expression (3.15) improve in accuracy as j increases for a constant i. Equation (3.15) is by far the best of the three for all values of m and p, except when m and p are large and nearly equal. In this range and only in this range is Equation (3.17) better.

3.7 Conclusions

The uniform reference model has been extensively studied in the literature because of its simplicity. However, it does not provide a good approximation to the performance of real-life systems in which programs have strong locality of reference. The local reference model proposed in this chapter explicitly models this property which characterizes a majority of real-life computer programs. Our results show that the performance of such programs in a multiprocessor system is significantly worse than what is predicted by the uniform reference model used earlier. It would thus be worthwhile to make serious efforts in designing programs with uniformly random addressing patterns. Efforts have been made (such as [Fer 76]) in the context of multiprogramming environments to make programs more local in behavior. The opposite of this is needed for multiprocessors. Research in the design of program with uniform addressing patterns could give valuable results and would help in improving substantially the performance of a multiprocessor syste

CHAPTER 4

MODEL FOR TIME-SHARED BUS SYSTEMS

In this chapter, we present a discrete Markov chain model for a multiprocessor with a time-shared bus as shown in Figure 3. After describing our notations in Section 4.1, we discuss the major assumptions of the model in Section 4.2. The analysis of the model is outlined in Section 4.3. An example is given in Section 4.4 to explain the analysis technique in detail. Section 4.5 presents simulation results for the example of Section 4.4 to show the accuracy of the analytical results.

4.1 Notations

We shall consider a multiprocessor system with p processors and m memory modules. The memory modules are of two types as explained in the assumptions to follow in the next section; there are m_1 memories of type 1 and m_2 $(= m - m_1)$ memories of type 2. The processing time of a processor will be denoted by t_p, the cycle times of the two types of memories by t_{c1} and t_{c2}, and the bus cycle time by t_b. The symbols α and $\beta (= 1 - \alpha)$ stand for the parameters of the processing time distribution while γ_1, $\delta_1 (= 1 - \gamma_1)$, γ_2, and $\delta_2 (= 1 - \gamma_2)$ are parameters of the memory cycle time distributions. The average processing time and the average memory cycle times are $\bar{t}_p$, $\bar{t}_{c1}$ and $\bar{t}_{c2}$ respectively. The unit instruction execution rates of instructions executed from the two types of memories are denoted by UER1 and UER2, the utilization factors of these memories by ρ_1 and ρ_2, and the total unit instruction execution rate of the system by UER. The states of the system will be represented by vectors whose components are denoted by k_0, k_1, k_2,

$k_{11}, k_{12}, \ldots, k_{21}, k_{22}, \ldots$, etc. The element T(i,j) of the matrix T stands for the transition probability from state s_i to state s_j. The equilibrium probability of state s_i is denoted by z(i).

4.2 Assumptions of the Model

The following major assumptions characterize the model developed in this chapter:

Assumption 1: The p processors and m memory modules of the system are connected by a time-shared bus with constant cycle time t_b. In one cycle time, the bus transfers information from a processor to a memory or vice versa.

Assumption 2: The processing times of all processors are identically geometrically distributed with the distribution

$$\Pr[t_p = i.t_b] = \beta \cdot \alpha^i, \quad i = 0,1,2, \ldots,$$

where $\beta = 1 - \alpha$.

The mean processing time is given by $\bar{t}_p = \alpha.t_b/\beta$. Thus any mean value of t_p can be considered by appropriately choosing α. That this is a realistic assumption has been shown by the models of Bhandarkar [Bha 75] who also argues that the processing time of a real-life processor has a distribution quite close to exponential.

Assumption 3: The memory modules are of two types: there are m_1 memories of type 1 and m_2 $(= m - m_1)$ memories of type 2. The access time of all memories is equal to their cycle time and the re-write time is zero (see Figure 6(b)). The cycle times of all memories of type j(j = 1,2) are identically geometrically distributed with the distribution

$$Pr[t_{cj} = i.t_b] = \delta_j \cdot \gamma_j^{i-1}, \quad i = 1,2, \ldots,$$

where $\delta_j = 1 - \gamma_j$.

The mean cycle time of a memory of type j is given by $\bar{t}_{cj} = t_b/\delta_j$. The difference between these distributions and the processing time distribution should be noted. The processing time takes on values 0, t_b, $2t_b$, $3t_b$, ..., whereas memory cycle times are not permitted to take value 0. As before, any mean value of t_{cj} can be chosen by appropriately choosing γ_j.

Assumption 3 makes our model more general than one which has all memory modules identical. The reason for choosing two different types of memories is that such a model will be needed for analysing the Pluribus system in the next chapter, where local memories and global memories may have different cycle times. Note that we can always choose $\bar{t}_{c1} = \bar{t}_{c2}$ (or $m_2 = 0$) as a particular case.

Assuming the memory cycle time to have a geometric distribution has been done to keep the analysis of the model tractable. We do not want to claim that this is a realistic assumption. Memory modules generally do have a constant cycle time. However, the model is robust enough for this assumption to make no significant difference. This will be borne out by the simulation results presented in Section 4.5 to validate this model.

Assumption 4: All processor requests to memory are read operations. The processor transmits the read address over the bus to the memory module. A memory module can fetch only one word in one cycle. Once the data has been fetched, it is transmitted back over the bus to the requesting processor. However, the memory module is not held up while the data is being transmitted and is free to take up a new request as soon as the previous cycle is over.

Assumption 5: The requests of any one processor are uniformly distributed over all memory modules, regardless of their type.

Assumption 6: Each memory module has a queue in which processor requests are queued. After a request has been served, the next request is taken from the queue if it is not empty.

Assumption 7: The time-shared bus has two queues, one each for the traffic in the two directions. The traffic from memories to processors (in queue 1) has priority over that from processors to memories (in queue 2).

The structure of this queueing model is shown in Figure 9.

4.3 Analysis of the Model

We shall now model the process defined by the assumptions of the previous section as a discrete Markov chain[1]. At any given time, the state of the system can be characterized by the lengths of the queues at the time-shared bus and the memory modules. This state is denoted by the vector

$$(k_0, k_1, k_2; k_{11}, k_{12}, \ldots, k_{1,m_1}; k_{21}, k_{22}, \ldots, k_{2,m_2})$$

where k_0 = number of processors in processing state; k_1 = length of queue 1 at the bus; k_2=length of queue 2 at the bus, k_{ji} = number of processors waiting in the queue of the i-th memory module of type j (including the processor being served), i = 1,2, ..., m_j, and j = 1,2. Note that k_1 or k_2 also includes the item being served by the bus, depending on the queue to which it belongs. The component k_0 of the vector is actually redundant

---

Footnote:

[1] For a description of Markov chain processes, see Kleinrock [Kle 75], and Bhandarkar [Bha 75].

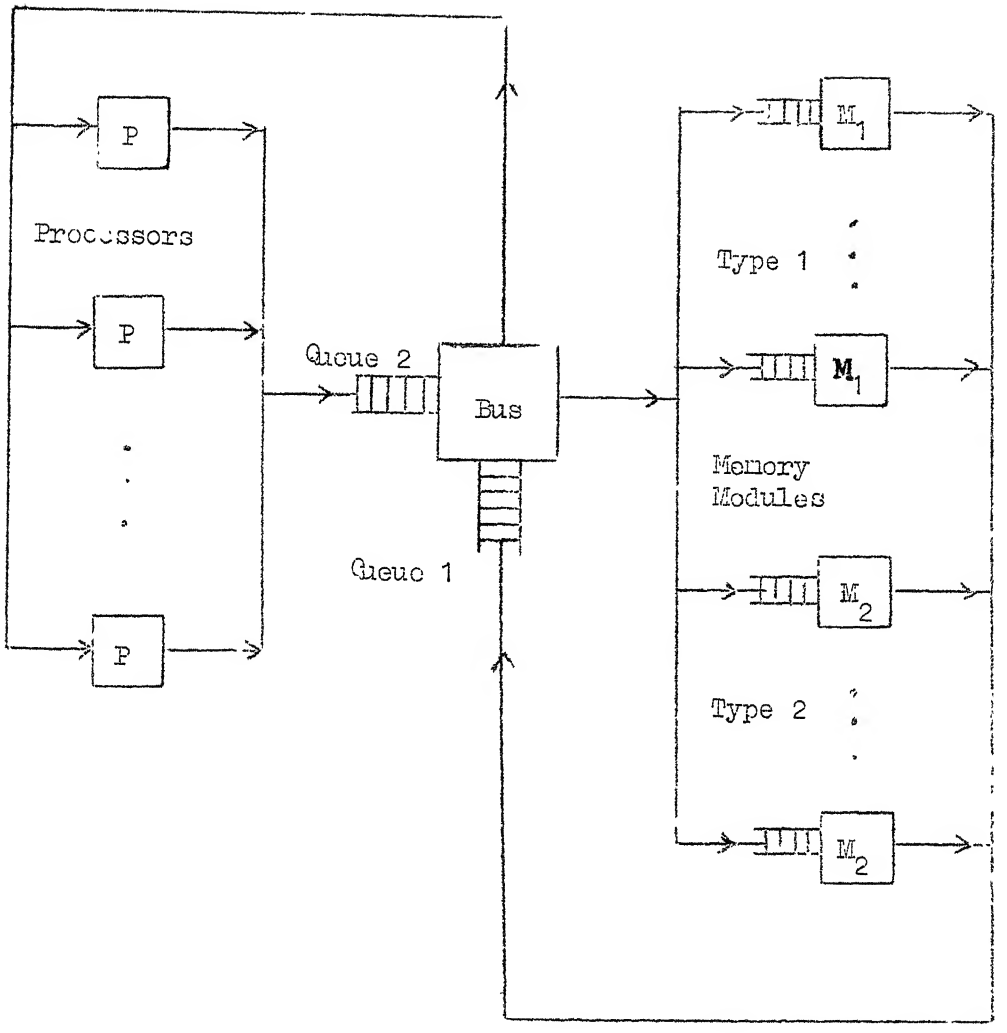


FIGURE 9: Structure of the queuing model for time-shared bus systems.

because

$$k_0 + k_1 + k_2 + \sum_{j=1}^{2} \sum_{i=1}^{m_j} k_{ji} = p;$$

it is included for the sake of convenience.

Since all processors are identical, and all memory modules of a particular type are also identical, a number of these states are equivalent, e.g., states (1,0,0; 1,2,0; 1,0), (1,0,0; 2,0,1; 1,0), (1,0,0; 1,2,0; 0,1), (1,0,0; 2,0,1; 0,1), etc. Every such equivalence class will be called a reduced state.

State changes occur only at the end of bus cycles. Thus if we consider transitions between states at points just prior to the end of bus cycles, we get a discrete-parameter Markov chain. The procedure for analysing this model is as follows:

Given the values of $p, m_1, m_2, \alpha, \gamma_1$ and γ_2, all the reduced states must be enumerated and the transition probabilities $T(i,j)$ between every pair of states (s_i, s_j) must be found. The method of enumerating the reduced states and finding the transition probabilities is similar to that employed in the crossbar switch models of the previous chapter; a systematic technique for it has also been given by Bhandarkar [Bha 75]. The equilibrium state probabilities $z(i)$ can now be evaluated by solving the set of equations

$$z(i) = \sum_j z(j) \cdot T(j,i) \qquad (4.1)$$

together with

$$\sum_i z(i) = 1 \qquad (4.2)$$

The unit instruction execution rate (UER) or any other performance measure one is interested in can be calculated from a knowledge of the equilibrium state probabilities.

This outline of the analysis method will be explained in detail by the example of the next section.

4.4 An Example

Consider a system with 2 processors and 3 memory modules, 2 of which belong to type 1 and one is of type 2. Thus $p = 2$, $m_1 = 2$, and $m_2 = 1$. For this case, there are 16 reduced states:

$$s_1 = (2,0,0;\ 0,0;\ 0), \qquad s_2 = (1,1,0;\ 0,0;\ 0),$$
$$s_3 = (1,0,1;\ 0,0;\ 0), \qquad s_4 = (1,0,0;\ 1,0;\ 0),$$
$$s_5 = (1,0,0;\ 0,0;\ 1), \qquad s_6 = (0,2,0;\ 0,0;\ 0),$$
$$s_7 = (0,1,1;\ 0,0;\ 0), \qquad s_8 = (0,1,0;\ 1,0;\ 0),$$
$$s_9 = (0,1,0;\ 0,0;\ 1), \qquad s_{10} = (0,0,2;\ 0,0;\ 0),$$
$$s_{11} = (0,0,1;\ 1,0;\ 0), \qquad s_{12} = (0,0,1;\ 0,0;\ 1),$$
$$s_{13} = (0,0,0;\ 2,0;\ 0), \qquad s_{14} = (0,0,0;\ 1,1;\ 0),$$
$$s_{15} = (0,0,0;\ 1,0;\ 1), \qquad s_{16} = (0,0,0;\ 0,0;\ 2).$$

The transition matrix for these states is given in the Appendix in terms of parameters α, γ_1, γ_2, and $\beta = 1 - \alpha$, $\delta_1 = 1 - \gamma_1$, $\delta_2 = 1 - \gamma_2$. A computer program can now be written which, given values for these parameters will solve the set of simultaneous linear equations (4.1), (4.2), and compute the equilibrium state probabilities $z(i)$. The unit instruction execution rate (UER) can now be found in the following way.

During any given period of time, every instruction executed by the system keeps the bus busy for two cycles: once for transferring the address to the memory, and again for transmitting the data back to the processor. Hence the sum of the equilibrium probabilities of those states during which the bus is serving an item from queue 1 will give the average number of instructions executed in one bus cycle of duration t_b. These states are s_2, s_6, s_7, s_8, and s_9. Alternatively, we may find the sum of the equilibrium probabilities of those states during which the bus is serving an item from queue 2. Since queue 1 has priority over queue 2, these states are s_3, s_{10}, s_{11}, and s_{12}. Thus we get,

$$\begin{aligned} UER &= (z(2) + z(6) + z(7) + z(8) + z(9))/t_b \\ &= (z(3) + z(10) + z(11) + z(12))/t_b. \end{aligned}$$

We are also interested in finding UER1 and UER2, the average rates of instructions executed from memories of type 1 and from memories of type 2 respectively. Clearly, UER = UER1 + UER2.

The sum of the equilibrium probabilities of all states weighted by the number of memories of type j busy during that state gives the utilization factor ρ_j of those memories. Thus

$$\rho_1 = z(4) + z(8) + z(11) + z(13) + 2.z(14) + z(15)$$

and $$\rho_2 = z(5) + z(9) + z(12) + z(15) + z(16).$$

Since the average cycle time for memories of type j is $\bar{t}_{cj} = t_b/\delta_j$, and one instruction is fetched by one memory module in one cycle, the average rate of instructions executed from memories of type j is

$$UERj = \rho_j/\bar{t}_{cj} = \rho_j \, \delta_j/t_b.$$

TABLE III

RESULTS FOR TIME-SHARED BUS MODEL

$p = 2$, $n_1 = 2$, $n_2 = 1$, $\bar{t}_{c1} = 600$ nsec.

$\bar{t}_p$ nsec	t_b nsec	$\bar{t}_{c2}$ nsec	Analytic Results			Simulation Results	Percentage Error
			UER1 insts/ μsec	UER2 insts/ μsec	UER insts/ μsec	UER insts/ μsec	
120	150	300	1.2514	0.6257	1.8772	1.928	2.71
		600	1.1309	0.5655	1.6964	1.785	5.22
		900	1.0193	0.5097	1.5290	1.593	4.19
	300	300	0.9122	0.4561	1.3683	1.394	1.88
		600	0.8514	0.4257	1.2772	1.337	4.68
		900	0.7926	0.3963	1.1889	1.232	3.63
240	150	300	1.1455	0.5728	1.7183	1.758	2.31
		600	1.0433	0.5217	1.5658	1.624	3.77
		900	0.9484	0.4742	1.4226	1.473	3.54
	300	300	0.8630	0.4315	1.2945	1.304	0.73
		600	0.8070	0.4035	1.2105	1.248	3.10
		900	0.7535	0.3767	1.1302	1.165	3.08
360	150	300	1.0508	0.5254	1.5763	1.621	2.84
		600	0.9643	0.4822	1.4465	1.482	2.45
		900	0.8834	0.4417	1.3251	1.382	4.29
	300	300	0.8130	0.4065	1.2195	1.235	1.27
		600	0.7626	0.3813	1.1439	1.176	2.81
		900	0.7146	0.3573	1.0719	1.095	2.16

TABLE III (Continued)

480	150	300	0.9681	0.4841	1.4522	1.479	1.85
		600	0.8944	0.4472	1.3416	1.386	3.31
		900	0.8251	0.4126	1.2377	1.282	3.58
	300	300	0.7657	0.3829	1.1486	1.150	0.12
		600	0.7207	0.3603	1.0810	1.097	1.48
		900	0.6777	0.3388	1.0165	1.032	1.52
600	150	300	0.8960	0.4480	1.3441	1.385	3.04
		600	0.8328	0.4164	1.2492	1.279	2.39
		900	0.7730	0.3865	1.1595	1.175	1.34
	300	300	0.7221	0.3610	1.0831	1.083	0.01
		600	0.6818	0.3409	1.0227	1.045	2.18
		900	0.6433	0.3217	0.9650	0.976	1.14
1200	150	300	0.6472	0.3236	0.9709	0.955	1.64
		600	0.6143	0.3071	0.9214	0.931	1.04
		900	0.5824	0.2912	0.8736	0.885	1.30
	300	300	0.5552	0.2776	0.8328	0.828	0.58
		600	0.5312	0.2656	0.7968	0.818	2.66
		900	0.5080	0.2540	0.7620	0.766	0.52
1800	150	300	0.5040	0.2520	0.7560	0.751	0.66
		600	0.4841	0.2420	0.7261	0.727	0.12
		900	0.4646	0.2323	0.6970	0.682	2.15
	300	300	0.4477	0.2239	0.6716	0.685	2.00
		600	0.4321	0.2161	0.6482	0.640	1.27
		900	0.4169	0.2085	0.6254	0.625	0.86

CE... RY
Acc. 54882

TABLE III (Continued)

2400	150	300	0.4119	0.2060	0.6179	0.608	1.60
		600	0.3987	0.1994	0.5981	0.595	0.52
		900	0.3857	0.1928	0.5785	0.575	0.61
	300	300	0.3742	0.1871	0.5612	0.542	3.42
		600	0.3633	0.1816	0.5449	0.548	0.57
		900	0.3526	0.1763	0.5290	0.536	1.32
3000	150	300	0.3481	0.1740	0.5221	0.527	0.94
		600	0.3386	0.1693	0.5080	0.505	0.59
		900	0.3294	0.1647	0.4940	0.490	0.81
	300	300	0.3210	0.1605	0.4815	0.483	0.31
		600	0.3130	0.1565	0.4695	0.475	1.17
		900	0.3052	0.1526	0.4578	0.447	2.36

Table III shows the values of UER, UER1 and UER2 computed from this model for $\bar{t}_{c1}$ = 600 nsec; and various values of $\bar{t}_p$, $\bar{t}_{c2}$, and t_b. A close observation of this table shows that

$$UERj = UER \cdot m_j/(m_1 + m_2).$$

This actually follows from Assumption 5, namely that the processors' requests are uniformly distributed over all memory modules. The difference in speed of the two types of memories will affect the total instruction execution rate, but each memory module will continue to receive its share of processor requests.

4.5 Simulation Results

Simulation studies were conducted to validate this model for time-shared bus systems. The simulation program assumed constant memory cycle times so that we could estimate the effect of Assumption 3 (geometrically distributed memory cycle times). The program was written in FORTRAN IV and run on an IBM 7044. To find the steady-state system performance, the instruction execution rate was averaged over a total of 5000 memory cycles. The results are shown in Table III along with the analytical results. It can be seen that in no case does the error exceed 6 percent. Thus we can say that the assumption of geometrically distributed memory cycle times is acceptable. The closeness of the analytical and simulation results demonstrates the usefulness of the model.

CHAPTER 5

MODEL FOR PLURIBUS SYSTEMS

In this chapter, we develop a model to analyse the performance of the Pluribus multiprocessor system which was described in Chapter 2. The notations used in this chapter are described in Section 5.1. The major assumptions of the model are outlined in Section 5.2. Section 5.3 gives the detailed analysis of the model. The analytic results are presented in Section 5.4 and are compared with the simulation results in Section 5.5.

5.1 Notations

For the Pluribus model in this chapter, we shall consider the system configuration shown in Figure 10. We assume the following parameters for this model:

n_p = number of processors on each processor bus,

n_{nl} = number of local memory modules on each processor bus,

n_{ng} = number of global memory modules,

n_{pb} = number of processor buses,

t_p = processing time of each processor,

t_{cl} = cycle time of each local memory module,

t_{cg} = cycle time of each global memory module,

and t_b = bus cycle time.

It should be noted that the system modelled here (Figure 10) differs in one respect from the actual configuration of the Pluribus system shown in Figure 5. The crossbar switch, instead of connecting to memory buses,

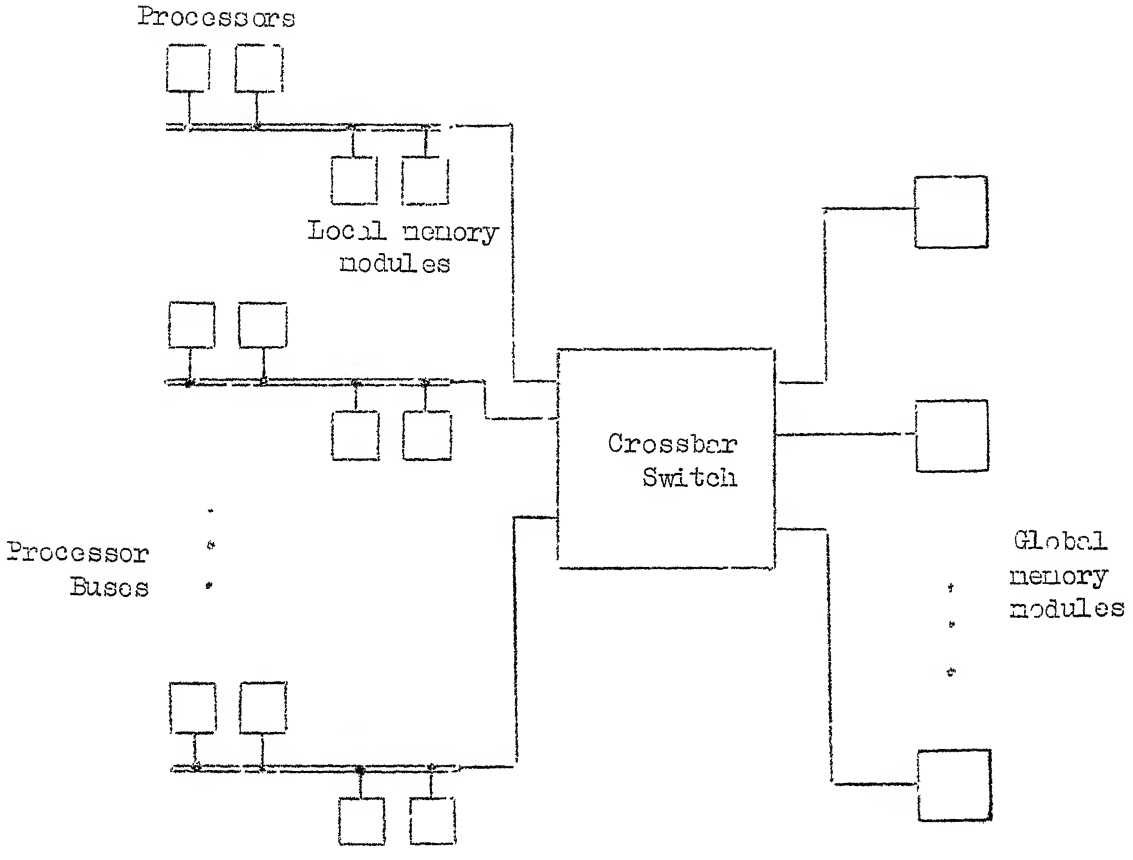


FIGURE 10: Pluribus configuration for the model.

here connects directly to global memory modules.

These two systems may be considered equivalent through the following transformation : Let the system in Figure 5 have n_{mb} memory buses with bus cycle time t_{mb} and each having n_{bg} memory modules. Let the cycle time of these memory modules be t_{cmg}. This system is then equivalent to the one in Figure 10 where the number of global memory modules is $n_{mg} = n_{mb} \cdot n_{bg}$ with memory cycle time $t_{cg} = t_{cmg} + 2t_{mb}$. The term $2t_{mb}$ in the cycle time reflects our assumption that two bus cycles are necessary to access one word from the memories: one for transferring the address to the module and the other for transmitting the data back to the processor.

This transformation is approximate, however. It neglects the queuing delays due to conflicts for the memory buses. If the number of memories per bus is not large and the buses are fast enough, this approximation is justified.

5.2 Assumptions of the Model

The following major assumptions will be made for this model:

Assumption 1: The crossbar switch has zero delay. However, crossbar switches with nonzero delay may be modelled by simply adding the delay to t_{cg}, the global memory cycle time.

Assumption 2: For all memory modules, local as well as global, the access time is equal to the cycle time and the rewrite time is zero. (See Figure 6(b)).

Assumption 3: Any particular processor can access the local memory modules on its own bus as well as all the global memory modules. These accesses have the following distribution : Each local memory module is accessed with probability $1/(n_{ml} + 1)$. Each global memory module is accessed with probability $1/(n_{mg}(n_{ml} + 1))$. There are two reasons for assuming this distribution: First, in any computer system where memory is divided into local and global parts, accesses from local memory are generally more frequent than those from global memory. The distribution we have assumed conforms to this pattern: the total probability assigned to accesses from all the global memory modules is equal to the probability assigned to accesses from one local memory module. The second reason is that, as will be clear in the next section, this distribution will permit us to use unchanged the bus model developed in chapter 4.

5.3 Analysis of the Model

We now give a method for analysing the performance of this model of the Pluribus system. We have seen in Chapter 3 that there are a number of models for analysing the crossbar switch system. We have also developed, in Chapter 4, an analytic model for the time-shared bus system. Since the Pluribus is a composite of a crossbar switch and time-shared buses, we shall try to decompose it into these components and use the models for these to analyse the performance of the whole. However, the bus and the crossbar subsystems are not isolated; there is an interaction between them and our model must take this into account. This is done in the following manner.

Consider a processor bus; let us call it B1. This bus interacts with the rest of the system (namely, the remaining processor buses, the crossbar switch, and the global memory modules) by sending requests for accessing the global memories. These requests are serviced after a certain time, which depends on the global memory cycle time and the interference due to requests from other processor buses. Thus, as far as bus B1 is concerned, the rest of the system simply looks and acts like a memory module with a certain cycle time, say t_{cv}. This is true for each of the processor buses. Hence the Pluribus system may be represented as shown in Figure 11. The system now consists of n_{pb} independent processor buses, each with n_p processors with processing time t_p, n_{nl} local memories with cycle time t_{cl}, and a 'virtual' memory module[1] with cycle time t_{cv}, which replaces the rest of the system.

The problem is to determine the average value of this cycle time t_{cv} of each of these 'virtual' memories. To do this, let us look at the system from the viewpoint of the crossbar switch. As far as the crossbar switch is concerned, each processor bus simply acts as a processor. It sends a request for accessing data from global memory, and after the request is satisfied, takes a certain average processing time, say t_{pv}, before sending

---

Footnote:

[1] The term 'virtual' memory used here denotes the fact that this is not a real memory module and should not be confused with its conventional meaning in computer systems architecture.

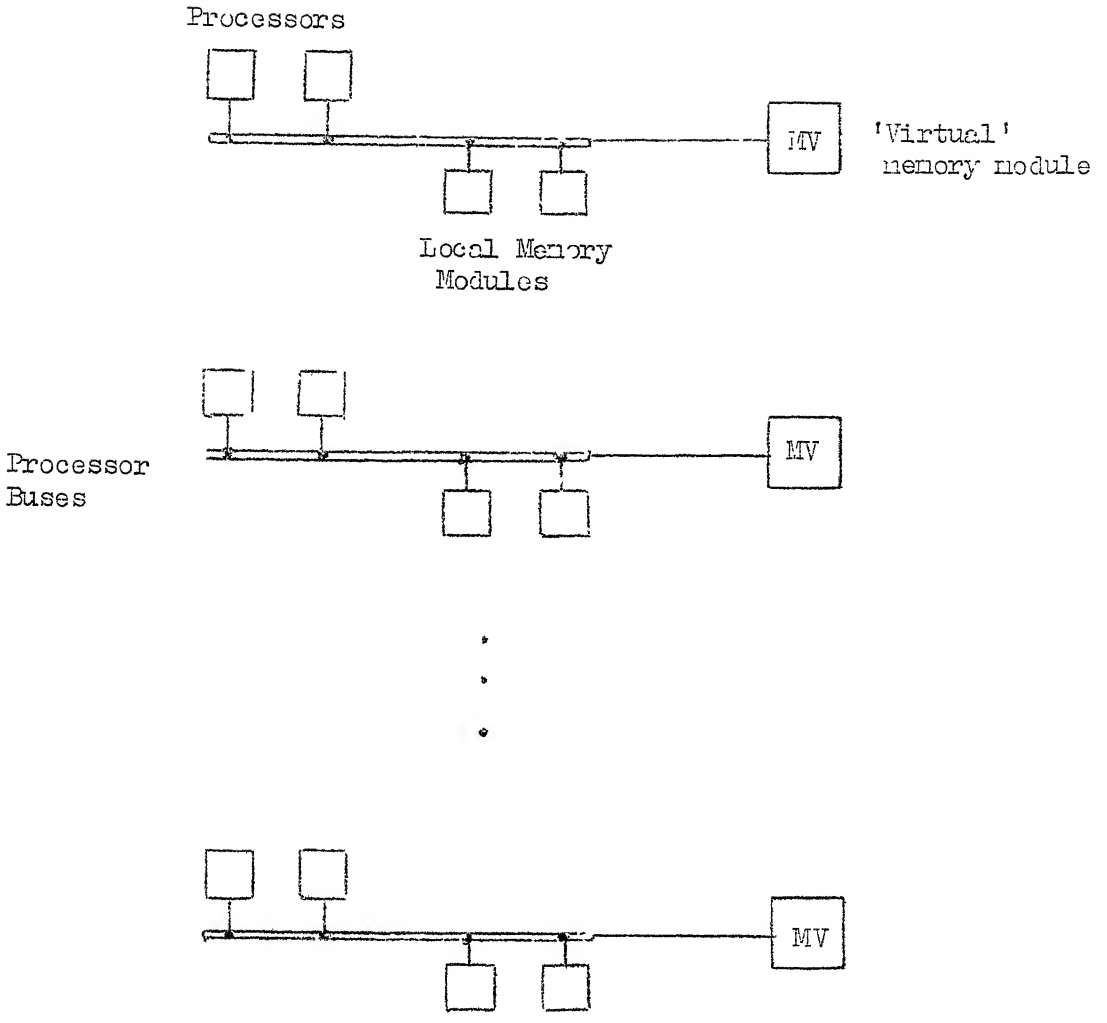


FIGURE 11: Pluribus system with 'virtual' memory modules (MV) replacing the crossbar switch component.

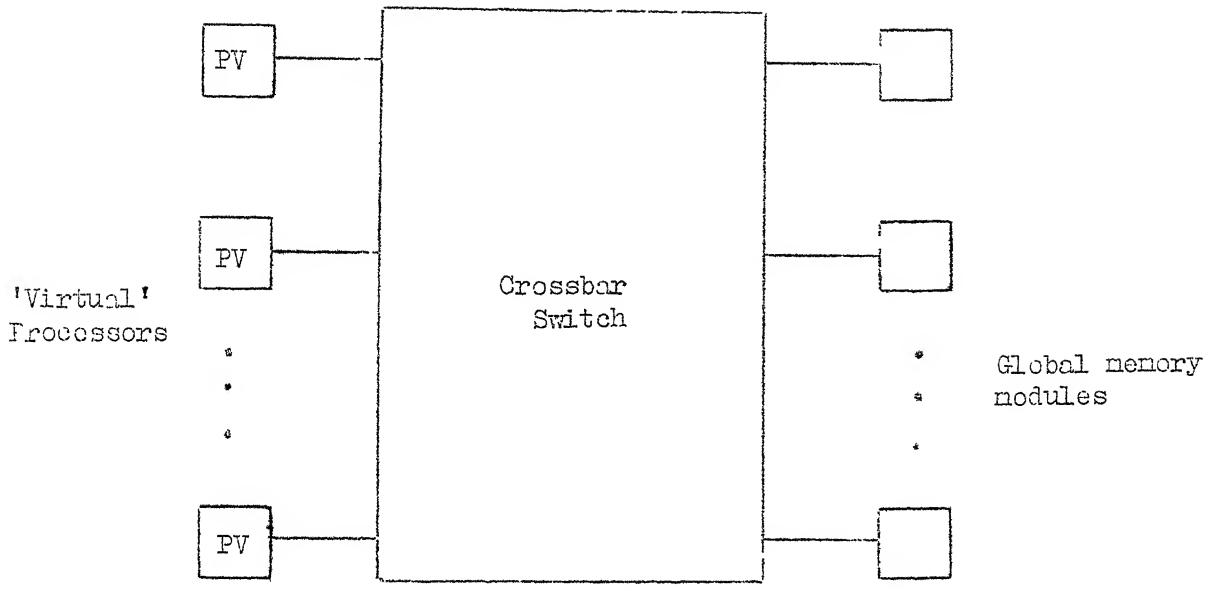


FIGURE 12: Crossbar component of the Pluribus system with 'virtual' processors (PV) replacing each processor bus.

FIGURE 13: Interaction between the bus and crossbar components.

global memories. In Figure 13, let $d(t_{pv}, t_{cv})$ be the rate at which the 'virtual' processor executes instructions from the 'virtual' memory. Clearly,

$$b(n_p, n_{ml}, t_p, t_{cl}, t_b, t_{cv}) = c(n_{pb}, n_{mg}, t_{cg}, t_{pv}) = d(t_{pv}, t_{cv}) \tag{5.1}$$

If the functions b,c, and d are known, then we have a set of two equations with two unknowns, t_{pv} and t_{cv}. These can be solved for and the performance of the total system is then found as follows.

As we noticed at the end of Section 4.4 of Chapter 4, for each processor bus, the rate of instruction execution from all memory modules (including the 'virtual' memory) is the same; it is thus given by $d(t_{pv}, t_{cv})$. Hence the total unit instruction execution rate for each bus is

$$(n_{ml} + 1) \cdot d(t_{pv}, t_{cv}).$$

Since there are n_{pb} buses in all, the total UER for the Pluribus system is

$$UER = n_{pb} \cdot (n_{ml} + 1) \cdot d(t_{pv}, t_{cv}) \tag{5.2}$$

It now remains to solve the system of Equations (5.1). In order to do this, we must know the functions b,c, and d. The function d is simply given by

$$d(t_{pv}, t_{cv}) = 1/(t_{pv} + t_{cv}) \tag{5.3}$$

For the crossbar switch, we may use Strecker's model [Str 70] described in Chapter 3. The function c is then given by Equation (3.1); this equation is repeated here:

$$c(n_{pb}, n_{ng}, t_{cg}, t_{pv}) = (n_{ng}/t_{cg})(1 - (1 - P_n/n_{ng})^{n_{pb}})$$

where

$$P_n + (n_{ng}/n_{pb})(t_{pv}/t_{cg})(1 - (1 - P_n/n_{ng})^{n_{pb}}) - 1 = 0 \qquad (5.4)$$

The value of this function can be computed for any given argument values by solving the above equation.

However, the function b is not known analytically. It can be computed for any given argument values by using the model developed in Chapter 4 (with $p = n_p$, $n_1 = n_{nl}$, $n_2 = 1$, $\bar{t}_p = t_p$, $\bar{t}_{c1} = t_{cl}$, $\bar{t}_{c2} = t_{cv}$, and $t_b = t_b$. The value of b is simply UER2).

Keeping into view the consideration that not all the functions are known analytically, it is possible to solve Equations (5.1) by the following iterative method.

Choose an initial value for t_{cv}. We shall choose $t_{cv} = t_{cg}$. Solve for t_{pv} from the equation

$$b(n_p, n_{nl}, t_p, t_{cl}, t_b, t_{cv}) = d(t_{pv}, t_{cv}) = 1/(t_{pv} + t_{cv})$$

i.e.,

$$t_{pv} = \frac{1}{b} - t_{cv}.$$

Using this value of t_{pv}, solve for t_{cv} from the equation

$$c(n_{pb}, n_{ng}, t_{cg}, t_{pv}) = d(t_{pv}, t_{cv}) = 1/(t_{pv} + t_{cv})$$

i.e.,

$$t_{cv} = \frac{1}{c} - t_{pv}.$$

One iteration is now completed. Use this value of t_{cv} to begin the next iteration. Repeat this process until the value of t_{cv} calculated at the end of an iteration is equal to the value of t_{cv} with which the iteration was begun.

The convergence of this process will, of course, depend on the functions b and c. These functions are not known analytically, so it is difficult to say anything definite about the convergence. Since the proof of the pudding is in its eating, we wrote a computer program to implement the complete method described in this section. The function b was calculated by solving numerically the set of simultaneous linear equations (4.1), (4.2). The function c was computed by numerically solving Equation (5.4) using the bisection method (it is known that the root P_n lies between 0 and 1). In the more than four hundred cases on which we tried this method, it never took more than 8 iterations, sometimes taking only 2 iterations. The FORTRAN IV program took about 2 minutes compilation time and execution time of the order of 2 seconds for each analysis on an IBM 7044. A few sample outputs are shown in Table IV.

5.4 Analytical Results

Since the Pluribus model has 8 input parameters, each of which can vary over a fairly wide range, it is impossible to obtain results covering the whole parameter space or even a significant part of it. For this reason, we limited ourselves to investigating the parameter space in the vicinity of the values of the actual Pluribus system built by BBN. This system is characterized by the following parameter values: $n_p = 2$, $n_{nl} = 2$, $n_{ng} = 4$, $n_{pb} = 7$, $t_p = 1425$ nsec., $t_{cl} = 850$ nsec., $t_{cg} = 1250$ nsec., $t_b = 200$ nsec.

TABLE IV

SAMPLE OUTPUTS OF ITERATIVE METHOD FOR ANALYSING PLURIBUS SYSTEM

Parameter Values (all times in nsec.)	No. of iteration	t_{cv} nsec	b insts/ μsec	t_{pv} nsec	c insts/ μsec	t_{cv} nsec	Total UER insts/ μsec
$n_p = 2$, $n_{nl} = 2$,	1	1250	0.2265	3166	0.2098	1602	
$n_{ng} = 4$, $n_{pb} = 7$,	2	1602	0.2155	3038	0.2149	1614	
$t_p = 1425$, $t_{cl} = 850$,	3	1614	0.2152	3033	0.2151	1615	
$t_{cg} = 1250$, $t_b = 200$.	4	1615	0.2151	3033	0.2151	1615	4.5180
$n_p = 2$, $n_{nl} = 2$,	1	850	0.2724	2821	0.2391	1361	
$n_{ng} = 2$, $n_{pb} = 7$,	2	1361	0.2505	2631	0.2479	1403	
$t_p = 1425$, $t_{cl} = 400$,	3	1403	0.2488	2616	0.2486	1407	
$t_{cg} = 850$, $t_b = 200$.	4	1407	0.2487	2615	0.2486	1407	5.2217
$n_p = 2$, $n_{nl} = 2$,	1	425	0.2539	3513	0.2523	450	
$n_{ng} = 4$, $n_{pb} = 5$,	2	450	0.2530	3502	0.2530	450	3.7957
$t_p = 1425$, $t_{cl} = 850$,							
$t_{cg} = 425$, $t_b = 200$.							

TABLE IV (CONTINUED)

n_p=2, n_{rl}=2,	1	1250	0.1965	3838	0.1878	1486	
n_{rg}=4, n_{pb}=6,	2	1486	0.1912	3745	0.1910	1492	
t_p=1425, t_{cl}=850,	3	1492	0.1910	3742	0.1911	1492	3.4388
t_{cg}=1250, t_b=400.							
n_p=2, n_{rl}=2,	1	1250	0.3059	2019	0.2396	2154	
n_{rg}=4, n_{pb}=10,	2	2154	0.2562	1749	0.2498	2254	
t_p=600, t_{cl}=850,	3	2254	0.2514	1724	0.2508	2264	
t_{cg}=1250, t_b=200.	4	2264	0.2509	1721	0.2509	2265	
	5	2265	0.2509	1721	0.2509	2265	7.5259
n_p=2, n_{rl}=2,	1	1000	0.2345	3264	0.1422	3769	
n_{rg}=1, n_{pb}=7,	2	3769	0.1613	2431	0.1428	4573	
t_p=1425, t_{cl}=850,	3	4573	0.1463	2263	0.1428	4740	
t_{cg}=1000, t_b=200.	4	4740	0.1434	2231	0.1428	4771	
	5	4771	0.1429	2226	0.1428	4776	
	6	4776	0.1428	2225	0.1428	4777	
	7	4777	0.1428	2224	0.1428	4778	
	8	4778	0.1428	2224	0.1428	4778	2.9990

Note that n_{ug} and t_{cg} are obtained by applying the transformation mentioned at the beginning of Section 5.1 (with $n_{ub} = 2$, $n_{bg} = 2$, $t_{ub} = 200$ nsec., $t_{cug} = 850$ nsec.) The value for t_p is the processing time per unit instruction (see definition of unit instruction in Section 2.5) and is obtained from the fact that the Lockheed SUE processors used by BBN have a 3.7 μsec add or load time (we took this to be a typical instruction) which is equivalent to two unit instructions containing two accesses from memory (with access time 425 nsec.)

These results are presented in Figure 14. The parts of this figure represent various cases of interest and are discussed below.

Figures 14(a) and (b): It is evident from these figures that no significant change in the performance is effected by changing the processor speed. For a constant processor speed, however, the performance increases almost linearly with the number of processor buses. This shows that the global memories do not constitute a bottleneck in the system. Note that these figures are for a system with 4 global memories. When the number of global memories is less, however, this statement does not hold, as we shall see below.

Figures 14(c) and (d): An observation similar to the above holds for the bus speed. These figures show that the effect of varying the bus speed is negligible as compared to the effect of varying the number of buses.

Figure 14(e): This is a very interesting figure. It shows that increasing the number of global memories from 1 to 4 improves the performance significantly. Beyond that, however, it is useless to further increase this number. On the other hand, increasing the global memory speed considerably

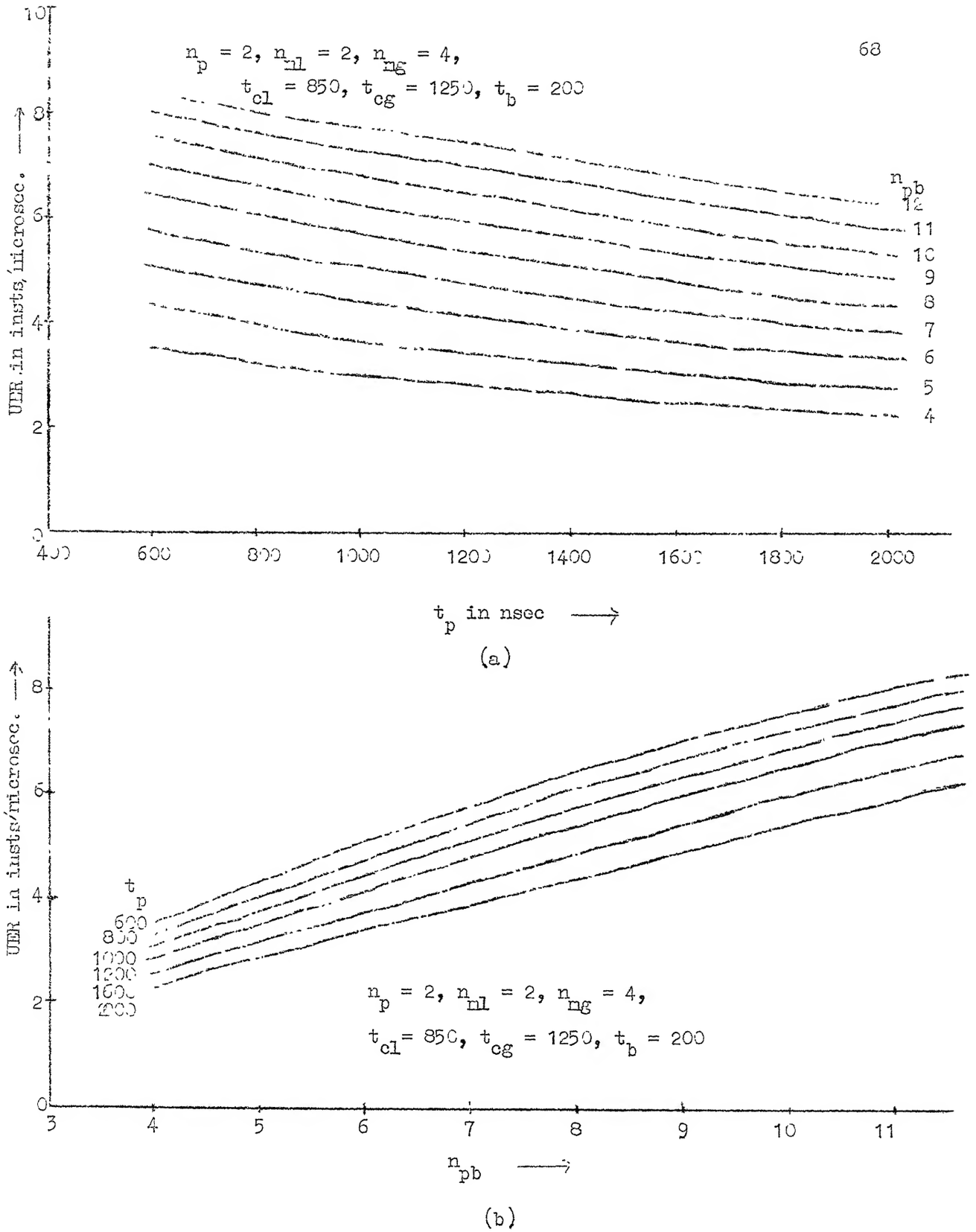
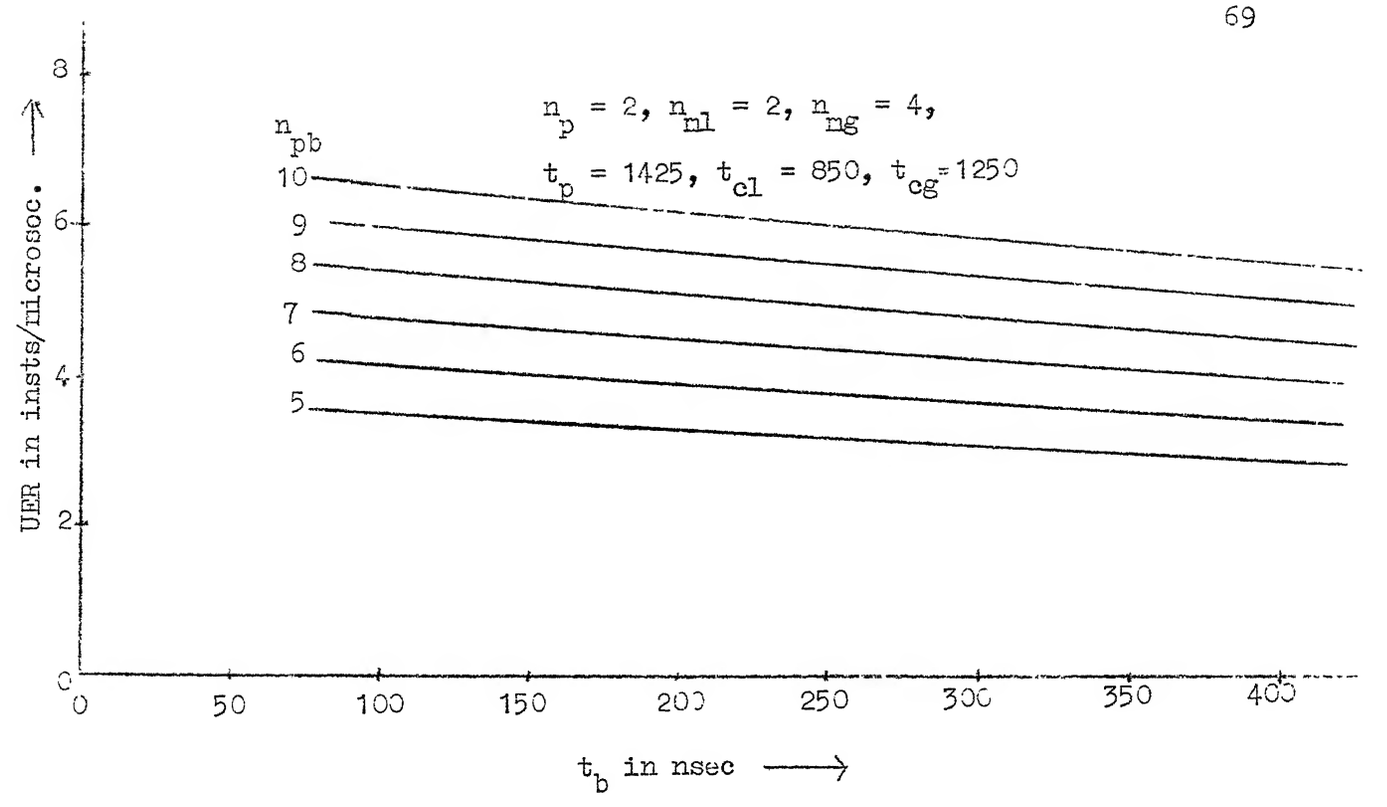
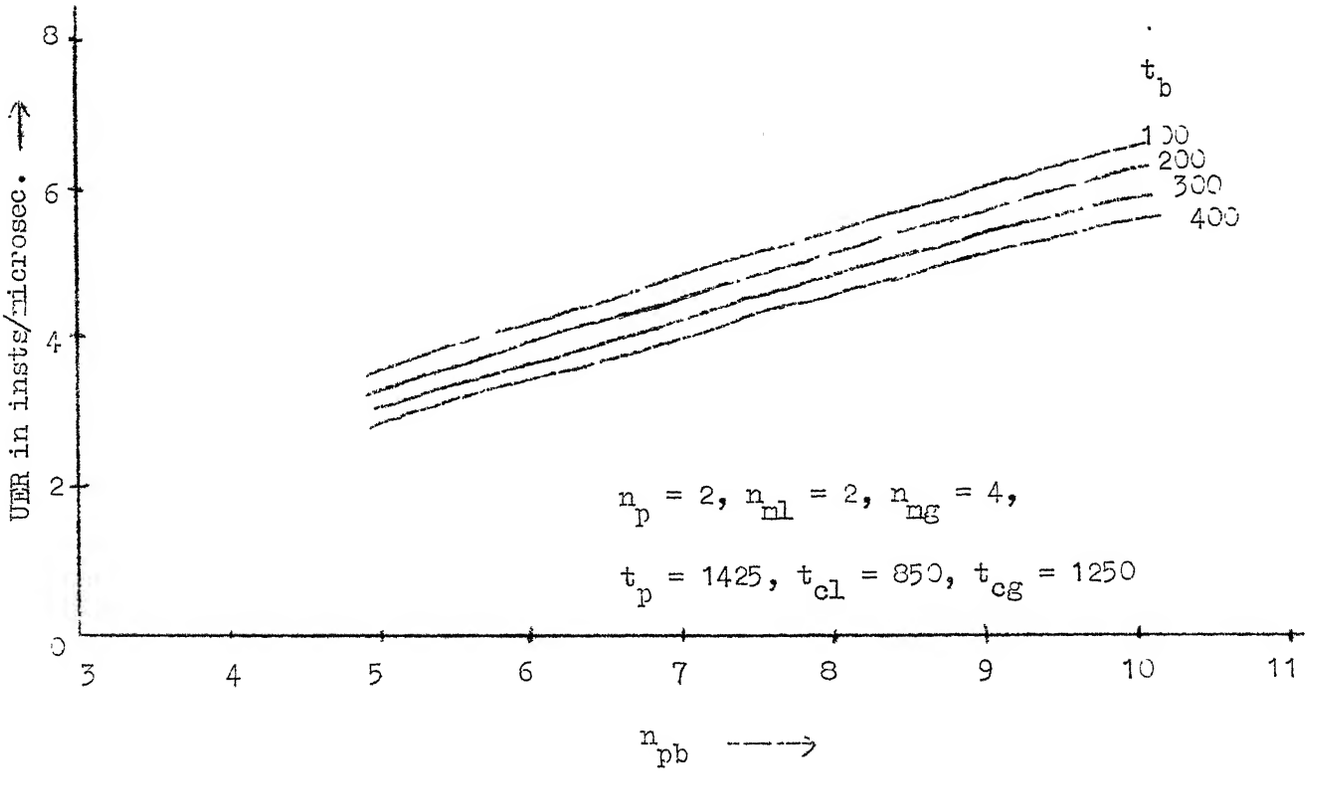


FIGURE 14: Analytical results for the Pluribus model.
(all times in nsec.)

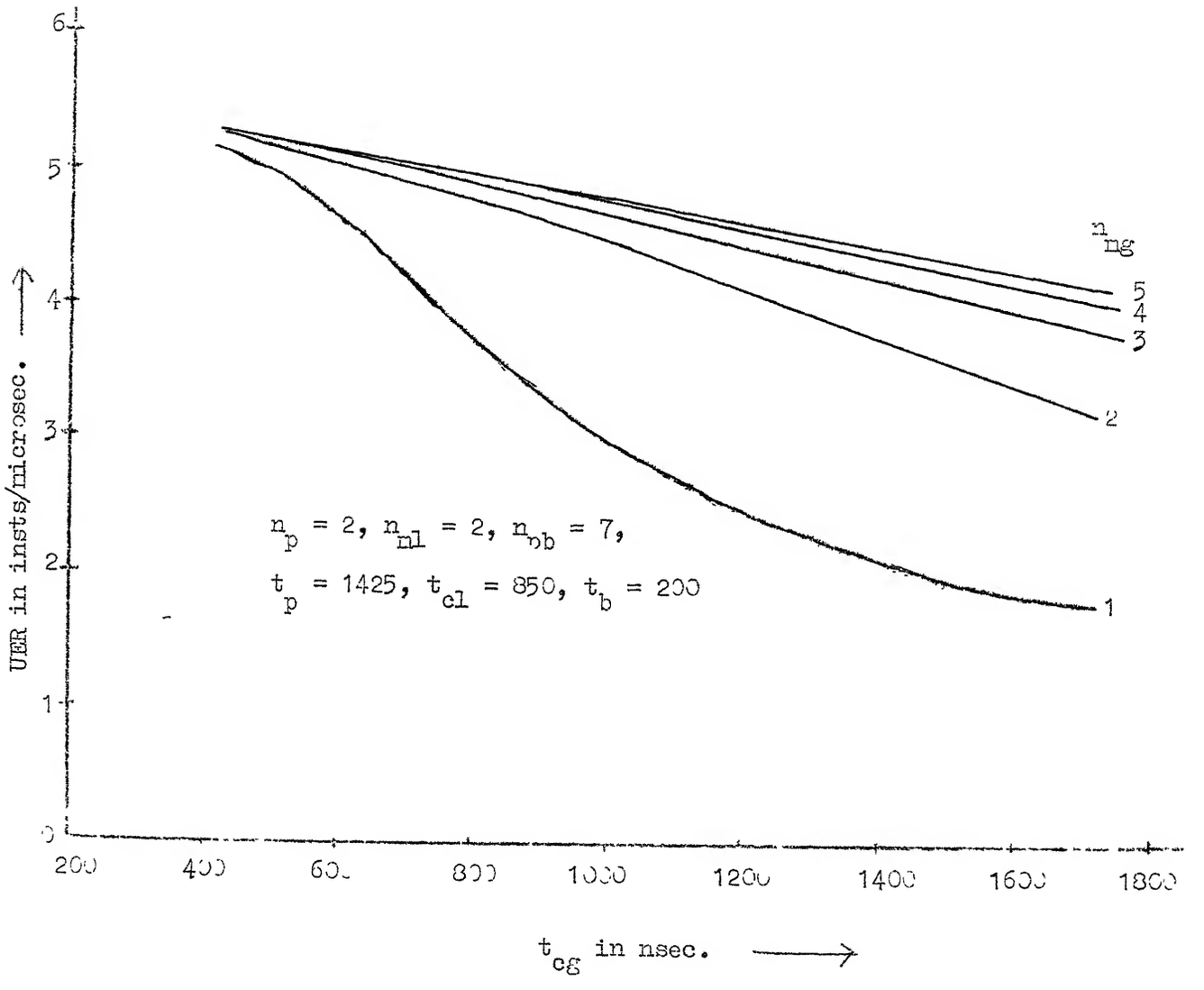


(c)



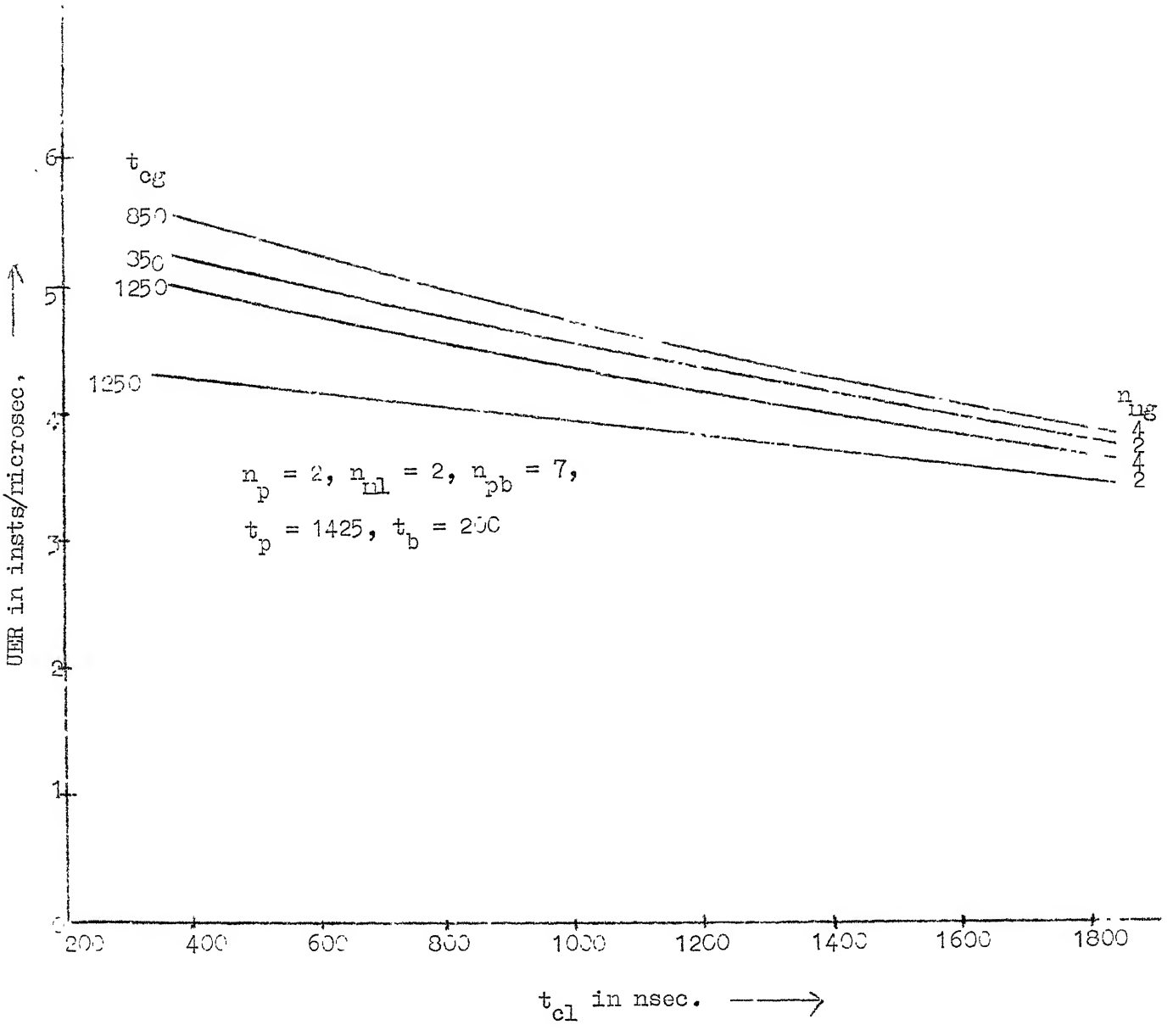
(d)

FIGURE 14: (Continued)
(all times in nsec.)



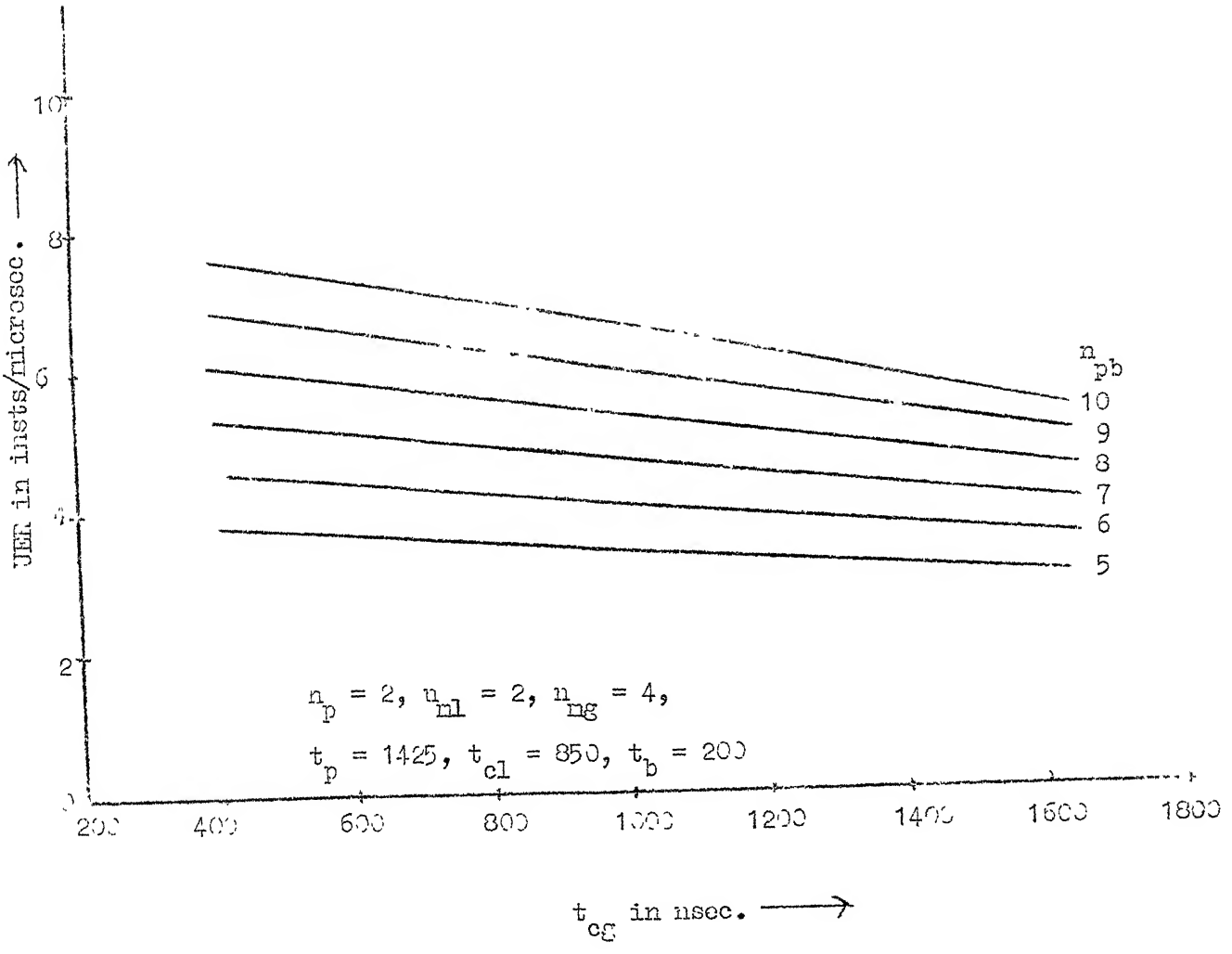
(e)

FIGURE 14: (Continued)
(all times in nsec.)



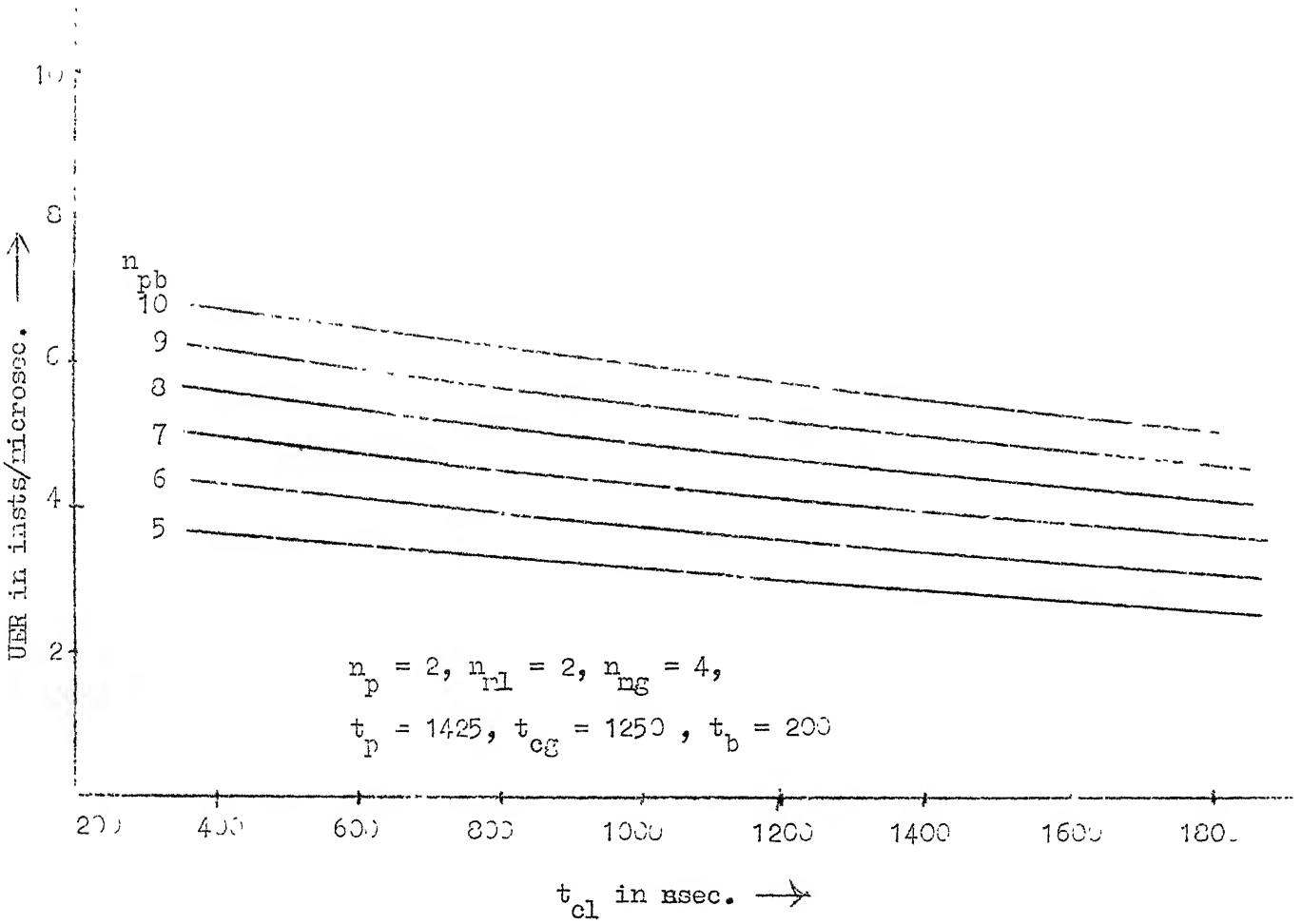
(f)

FIGURE 14: (Continued)
(all times in nsec.)



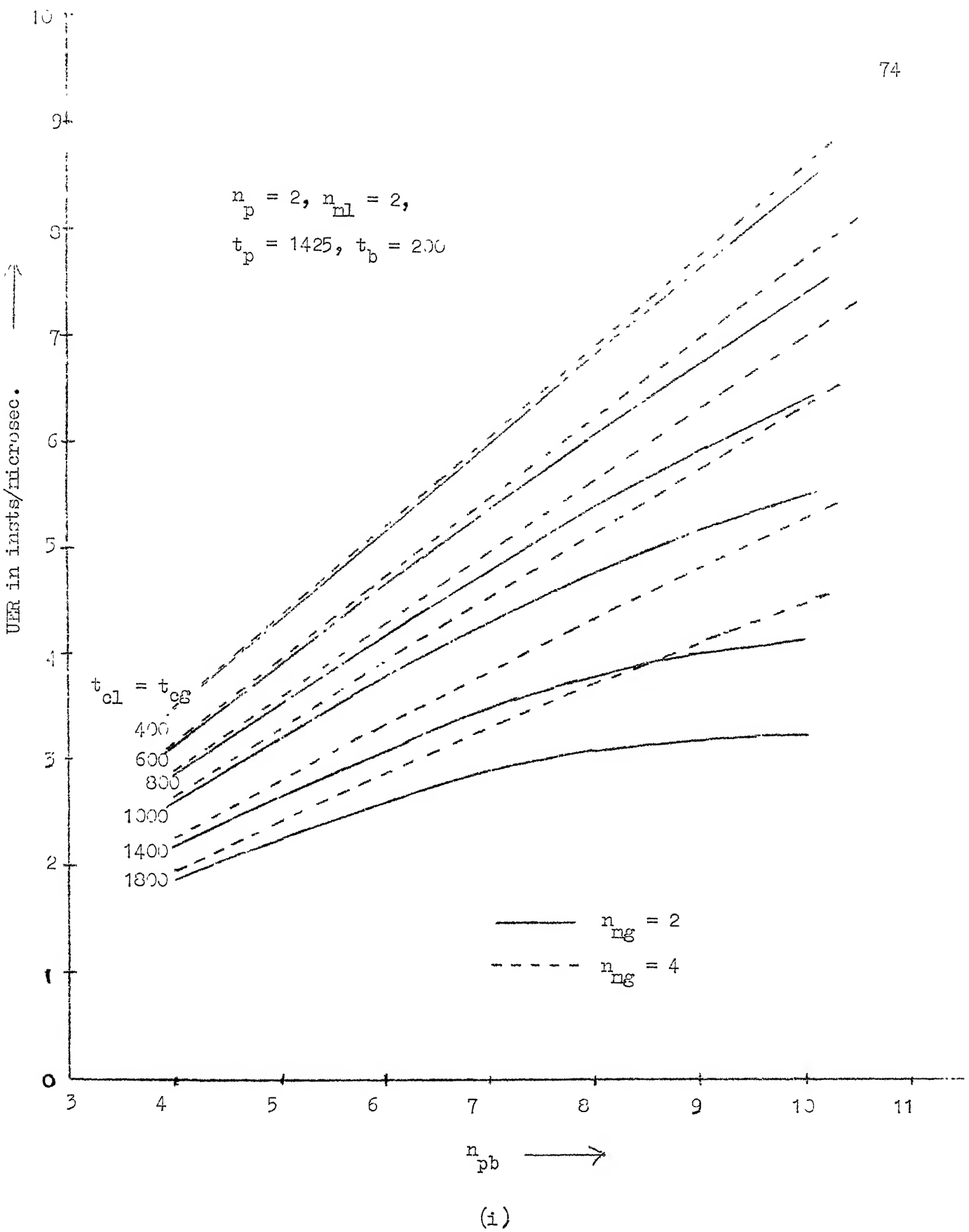
(g)

FIGURE 14: (Continued)
(all times in nsec.)



(h)

FIGURE 14; (Continued)
(all times nsec.)



(i)

FIGURE 14: (Continued)
(all times in nsec.)

improves the performance irrespective of the number of global memories.

Figure 14(f): This figure shows how the performance varies with the local memory speed for various global memory configurations. Clearly, 2 global memories with cycle time 850 nsec. are better than 4 of 1250 nsec. The local memory speed also has an impact on the performance.

Figure 14(g) and (h): These figures, together with the observations made earlier, show that, although faster global and local memories are better, the predominant effect on the performance is that of the number of processor buses. Figure 14(i), however, gives a better commentary on the interaction between these factors.

Figure 14(i): In this figure, the local and global memory speeds were kept equal. It is seen that for slower memories, with 2 global memories, the performance tends to saturate when the number of processor buses is increased. For faster memories, there is hardly any difference between the curves for 2 and 4 global memories. In this region, there is an almost linear performance increase with the number of buses.

Since the system built at BBN has 4 global memories, we can interpret these figures to mean that the system performance can be improved by speeding up the memories, but a spectacular linear improvement can be achieved by increasing the number of processor buses. However, increasing the number of global memories and increasing the processor and bus speeds will not have any significant impact on the performance. It also seems reasonable to assert that, just as in crossbar switch systems, the performance of the Pluribus can be increased without any law of diminishing returns so long as the processor-memory bandwidths are kept matched.

TABLE V

COMPARISON OF ANALYTIC AND SIMULATION RESULTS FOR PLURIBUS MODEL

$n_p = 2$, $n_{ml} = 2$

n_{mg}	n_{pb}	t_p nsec	t_{cl} nsec	t_{cg} nsec	t_b nsec	Analytic UER insts/ μsec	Simulation UER insts/ μsec	Percentage Error
1	7	1425	850	425	200	5.1394	5.2247	1.66
1	7	1425	850	1000	200	2.9990	3.0273	0.94
2	4	1425	400	400	200	3.4923	3.5740	2.34
2	4	1425	1000	1000	200	2.6235	2.7160	3.53
2	5	1425	500	500	200	4.1347	4.2296	2.30
2	5	1425	1200	1200	200	2.9519	3.0602	3.67
2	6	1425	600	600	200	4.6846	4.7943	2.34
2	6	1425	1400	1400	200	3.1497	3.2194	2.21
2	7	1425	400	850	200	5.2217	5.2705	0.93
2	7	1425	500	500	200	5.7337	5.8423	1.89
2	7	1425	500	1250	200	4.2306	4.1876	1.02
2	7	1425	850	500	200	5.1852	5.3191	2.58
2	7	1425	850	1200	200	4.1354	4.1471	0.28
2	7	1425	1000	850	200	4.5636	4.6654	2.23
2	7	1425	1200	1200	200	3.9112	3.9723	1.56
2	7	1425	1200	1250	200	3.8408	3.8837	1.12
2	8	1425	700	700	200	5.7890	5.8889	1.73
2	8	1425	1600	1600	200	3.4393	3.3811	1.69

TABLE V (Continued)

2	9	1425	800	800	200	5.9993	5.9862	0.22
2	9	1425	1800	1800	200	3.2232	3.1259	3.02
2	10	1425	400	400	200	8.5542	8.6715	1.37
2	10	1425	1000	1000	200	5.5554	5.3506	3.69
3	7	1425	850	700	200	5.0239	5.1593	2.70
3	7	1425	850	1400	200	4.2161	4.3191	2.44
4	4	600	850	1250	200	3.5119	3.7111	5.67
4	4	1400	850	1250	200	2.6792	2.7807	3.79
4	4	1425	500	500	200	3.3387	3.4220	2.50
4	4	1425	1200	1200	200	2.4617	2.5832	4.94
4	5	800	850	1250	200	4.0230	4.2322	5.20
4	5	1425	400	1250	200	3.6726	3.7750	2.79
4	5	1425	600	600	200	3.9717	4.0837	2.82
4	5	1425	850	425	200	3.7957	3.8960	2.64
4	5	1425	850	1000	200	3.4497	3.5675	3.41
4	5	1425	850	1250	100	3.5210	3.6529	3.75
4	5	1425	850	1250	250	3.1861	3.2998	3.57
4	5	1425	1000	1250	200	3.1783	3.3010	3.86
4	5	1425	1400	1400	200	2.8291	2.9790	5.30
4	5	1600	850	1250	200	3.1291	3.2344	3.37
4	6	1000	850	1250	200	4.4532	4.6649	4.75
4	6	1425	500	1250	200	4.2484	4.3916	3.37
4	6	1425	700	700	200	4.5342	4.6714	3.03
4	6	1425	850	500	200	4.4969	4.6144	2.61

TABLE V (Continued)

4	6	1425	850	1200	200	3.9549	4.0847	3.28
4	6	1425	850	1250	150	4.0436	4.2021	3.92
4	6	1425	850	1250	300	3.6685	3.7934	3.40
4	6	1425	1200	1250	200	3.6110	3.7622	4.19
4	6	1425	1600	1600	200	3.1233	3.2740	4.83
4	6	1800	850	1250	200	3.5238	3.6292	2.99
4	7	1200	850	1250	200	4.8198	5.0014	3.77
4	7	1425	600	600	200	5.5335	5.6743	2.54
4	7	1425	600	850	200	5.2525	5.3717	2.27
4	7	1425	700	1250	200	4.6734	4.8314	3.38
4	7	1425	850	700	200	5.0582	5.2028	2.86
4	7	1425	850	850	200	4.9162	5.0812	3.36
4	7	1425	850	1250	200	4.5180	4.6748	3.47
4	7	1425	850	1250	350	4.1100	4.2252	2.80
4	7	1425	850	1400	200	4.3653	4.5071	3.25
4	7	1425	850	1600	200	4.1623	4.3073	3.48
4	7	1425	1400	850	200	4.2730	4.4556	4.27
4	7	1425	1400	1400	200	3.8893	4.0834	4.99
4	7	1425	1600	1250	200	3.8272	3.9749	3.86
4	7	2000	850	1250	200	3.8727	3.9734	2.60
4	8	600	850	1250	200	6.4194	6.5819	2.53
4	8	1400	850	1250	200	5.1364	5.3035	3.25

TABLE V (Continued)

4	8	1425	600	1250	200	5.3841	5.5233	2.59
4	8	1425	800	800	200	5.7238	5.9070	3.20
4	8	1425	850	850	200	5.5904	5.7609	3.05
4	8	1425	850	1250	250	4.9489	5.0800	2.65
4	8	1425	850	1250	400	4.5145	4.6487	2.97
4	8	1425	850	1600	200	4.6600	4.8125	3.27
4	8	1425	1400	1250	200	4.5303	4.7251	4.30
4	8	1425	1800	1800	200	3.7738	3.9344	4.26
4	9	800	850	1250	200	6.6645	6.8668	3.04
4	9	1425	400	400	200	7.8388	8.0010	2.07
4	9	1425	700	1250	200	5.8394	5.9483	1.86
4	9	1425	850	425	200	6.8054	6.9755	2.50
4	9	1425	850	1000	200	6.0392	6.1993	2.65
4	9	1425	850	1250	100	5.9905	6.1348	2.41
4	9	1425	850	1250	300	5.3408	5.4739	2.49
4	9	1425	1000	1000	200	5.8283	6.0256	3.39
4	9	1425	1600	1250	200	4.8468	5.0061	3.29
4	9	1600	850	1250	200	5.4129	5.5169	1.92
4	10	1000	850	1250	200	6.8713	7.0482	2.57
4	10	1425	500	500	200	8.2539	8.4320	2.16
4	10	1425	800	1250	200	6.2566	6.3687	1.79
4	10	1425	850	500	200	7.4499	7.6367	2.51
4	10	1425	850	1200	200	6.2886	6.4082	1.90

TABLE V (Continued)

4	10	1425	850	1250	150	6.3647	6.4845	1.88
4	10	1425	850	1250	350	5.6975	5.8412	2.52
4	10	1425	1200	1200	200	5.8544	5.9969	2.43
4	10	1425	1800	1250	200	5.1311	5.3574	4.41
4	10	1800	850	1250	200	5.6562	5.7656	1.93
4	11	1200	850	1250	200	7.0501	7.0362	0.20
4	11	2000	850	1250	200	5.8728	5.9838	1.89
4	12	600	850	1250	200	8.3477	8.2118	1.63
4	12	1400	850	1250	200	7.2054	7.2179	0.17
5	7	1425	850	425	200	5.3104	5.4478	2.59
5	7	1425	850	1000	200	4.8073	4.9641	3.26
6	7	1425	850	500	200	5.2536	5.3944	2.68
6	7	1425	850	1200	200	4.6571	4.8268	3.64

5.5 Simulation Results

Simulation studies were conducted to validate the model for the Pluribus system presented in this chapter. The simulation program was written in FORTRAN IV and run on an IBM 7044. The program made only the three assumptions listed in Section 5.2; it took constant processing times as well as constant memory cycle times. The instruction execution rate was averaged over a total of 5000 memory cycles. This amounted to the processing of anywhere between 5000 and 32000 instructions (approximately) by the multiprocessor system, depending on the system parameters.

The simulation results together with the analytic results for some representative cases are shown in Table V. It can be seen that the errors are all below 6 percent. This verifies the accuracy and usefulness of the analytic Pluribus model. It also shows that the approximations made in replacing parts of the system by 'virtual' memories and 'virtual' processors (see Section 5.3) are reasonable and do not have a significant impact on the accuracy of the model.

CHAPTER 6

CONCLUSIONS

We shall now summarize the main results obtained in this thesis and suggest directions that future work in this area may take.

In Chapter 3, we discussed a model for crossbar switch systems which takes into account the local referencing property that characterizes most computer programs. It was found that the performance of multiprocessor systems with the local reference model is worse than that predicted by the traditional uniform reference model. We also derived new expressions for the uniform reference model and compared them with expressions available in the literature. Our expressions were more accurate than the existing expressions in most cases.

In Chapter 4, we presented a Markov chain model for multiprocessor systems using a time-shared bus. For reasons mentioned in Chapter 2, a single time-shared bus is not generally used in large multiprocessor systems. However, it is useful to have such models, not only because they help us in analysing other systems such as the Pluribus, but also because they aid in the understanding of evaluation techniques for multiprocessors.

In Chapter 5, we presented an analytic model for the Pluribus system. This model decomposes the Pluribus into its components comprising the crossbar switch and the processor buses. The performance of the total system is calculated in an iterative way from the performance of the two different components. In presenting our results, we used the model for the time-shared bus developed earlier and an expression derived by Strecker for the

crossbar switch. We would now like to point out that the iterative method used in the model is independent of the models or expressions that may be used for calculating the performances of the component systems. If, as seems plausible, better methods are available in future to analyse the time-shared bus and crossbar switch systems, these may be profitably used in this Pluribus model to give better prediction of the system performance. Further, if analytic expressions are found for both the sub-systems, it may even become possible to express the Pluribus performance analytically by solving Equation (5.1).

However, the absence of an analytical expression for the Pluribus performance in no way detracts from the usefulness of the model. The program written to implement the analysis method is quite simple and allows a system designer to quickly evaluate a large design space in an efficient way. This should help in evaluating the performance of systems based on the Pluribus architecture, and in pointing out where the bottlenecks lie and what methods to use for improving the system performance.

It is also our hope that our efforts will spark an interest in devising better analytic models for multiprocessor systems. The tools available in this area are still pitifully few and a lot of work is needed to catch up with the rapid growth in multiprocessor technology. Work on performance evaluation has so far been limited only to crossbar switch systems. We have now given analytic models for time-shared bus and Pluribus systems.

These models themselves stand in need of improvement. In addition, it is necessary to work on models for multiport memory/multibus systems and for other unconventional systems which are being designed these days. With the advent of microprocessors on the computer scene, it is now becoming feasible to construct multiprocessor systems containing a large number of processors (typically hundreds of microprocessors) which represents an increase of an order of magnitude in the number of functional units connected to the system. Clearly, the interconnection mechanism used in these systems is of crucial importance and work is in progress to devise new and better interconnection structures [Bar 75, Swa 76]. To keep pace with these developments, the tools of analytic modelling must be improved so that the system designer can easily evaluate and compare the various choices he faces. We hope that this thesis has been one step forward towards these goals.

REFERENCES

[And 75] Anderson, G.A., and Jensen, E.D., "Computer interconnection structures : taxonomy, characteristics, and examples," Computing Surveys 7,4 (Dec. 1975), 197-213.

[Bae 76] Baer, J.-L., "Multiprocessing Systems," IEEE Trans. Computers C-25,12 (Dec. 1976), 1271-1277.

[Bar 75] Barker, W.B., "A multiprocessor design," Ph.D. dissertation, Harvard Univ., Cambridge, Mass., Oct. 1975.

[Bas 76] Baskett, F., and Smith, A.J., "Interference in multiprocessor computer systems with interleaved memory," Comm. ACM, 19,6 (June 1976), 327-334.

[Bha 73] Bhandarkar, D.P., and Fuller, S.H., "Markov chain models for analyzing memory interference in multiprocessor computer systems," Proc. First Annual Symposium on Computer Architecture (Dec. 1973), Univ. of Florida, Gainesville, Fla.

[Bha 75] Bhandarkar, D.P., "Analysis of memory interference in multiprocessors," IEEE Trans. Computers C-24,9 (Sept. 1975), 897-908.

[Bha 76] Bhandarkar, D.P., "A hierarchy of analytic models for complex computer systems," Proc. European Computing Conferences on Computer Systems Evaluation (Sept. 1976), London, England.

[Bel 71] Bell, C.G., Broadley, W., Wulf, W., Newell, A., Pierson, C., Reddy, R., and Rege, S., "C.mmp : the CMU multiminiprocessor computer," Technical Report CMU-CS-72-112, Carnegie-Mellon Univ., Pittsburgh, Pa., Aug. 1971.

[Bur 70] Burnett, G.J., and Coffman, E.G., Jr., "A study of interleaved memory systems," in 1970 Spring Joint Computer Conf., AFIPS Conf. Proc. Vol. 36. AFIPS Press, Montvale, N.J. (1970), 467-474.

[Bur 73] Burnett, G.J., and Coffman, E.G., "A combinatorial problem related to interleaved memory systems," J.ACM 20,1(Jan. 1973), 39-45.

[Bur 75] Burnett, G.J., and Coffman, E.G., Jr., "Analysis of interleaved memory systems using blockage buffers," Comm. ACM 18,2 (Feb. 1975), 91-95.

[Cof 71] Coffman, E.G., Burnett, G.J., and Snowdon, R.A., "On the performance of interleaved memories with multiple-word bandwidth," IEEE Trans. Computers C-20,12 (Dec. 1971), 1570-1573.

[Ens 74] Enslow, P.H., ed., Multiprocessors and parallel processing, John Wiley, New York, 1974.

[Ens 77] Enslow, P.H., Jr., "Multiprocessor organization - a survey," Computing Surveys 9,1 (March 1977), 103-129.

[Fer 76] Ferrari, D., "The improvement of program behavior," Computer 9,11 (Nov. 1976), 39-47.

[Fer 73] Ferreira, R.C., and Rubino, A.V., "Architectural trends in minicomputers," in Infotech. State of the Art Report, Vol. 13 : Minicomputers. Infotech Information Ltd., Maidenhead, Berkshire, England, 1973, pp. 379-400.

[Fly 66] Flynn, M.J., "Very high-speed computing systems," Proc. IEEE 54,12 (Dec. 1966), 1901-1909.

[Fly 72a] Flynn, M.J., "Towards more efficient computer organizations," in 1972 Spring Joint Computer Conf., AFIPS Conf. Proc., Vol. 40. AFIPS Press, Montvale, N.J. (1972), 1211-1217.

[Fly 72b] Flynn, M.J., "Some computer organizations and their effectiveness," IEEE Trans. Computers C-21,9 (Sept. 1972), 948-960.

[Fly 72c] Flynn, M.J., and Podvin, A., "An unconventional computer architecture: shared resource multiprocessing," Computer 5,2 (Mar-Apr 1972), 20-28.

[Gor 67] Gordon, W.J., and Newell, G.F., "Closed queuing systems with exponential servers," Operations Research 15 (1967), 254-265.

[Hea 73] Heart, F.E., Ornstein, S.M., Crowther, W.R., and Barker, W.B., A new minicomputer/multiprocessor for the ARPA network," in 1973 National Computer Conf., AFIPS Conf. Proc., Vol. 42. AFIPS Press, Montvale, N.J. (1973), 529-537.

[Jac 63] Jackson, J.R., "Jobshop-like queuing systems," Management Science 10,1 (Oct. 1963), 131-142.

[Jan 71] Janson, P., "Common bus structure for minicomputers improves I-O flexibility," Control Engg. 18,1 (Jan. 1971), 50-53.

[Jos 76] Joseph, M., "Outline of a close-coupled multiprocessor system," Proc. Eleventh Annual Convention of Computer Society of India (1976), Hyderabad.

[Kle 75] Kleinrock, L., Queuing Systems, Vol. 1 : Theory, John Wiley, New York, 1975.

[Kle 76] Kleinrock, L., Queuing Systems, Vol. 2 : Computer Applications. John Wiley, New York, 1976.

[Nay 76] Nayak, P.V.S., and Jinega, B.C., "TDC 316 multiprocessor," Proc. Eleventh Annual Convention of Computer Society of India (1976), Hyderabad.

[Orn 75] Ornstein, S.M., Crowther, W.R., Kraley, M.F., Bressler, R.D., Michel, A., and Heart, F.E., "Pluribus - a reliable multiprocessor," in 1975 National Computer Conf., AFIPS Conf. Proc. Vol. 44, AFIPS Press, Montvale, N.J. (1975), 551-559.

[Rav 72] Ravi, C.V., "On the bandwidth and interference in interleaved memory systems," IEEE Trans. Computers C-21, 8 (Aug. 1972), 899-901.

[Sas 75] Sastry, K.V., and Kain, R.Y., "On the performance of certain multiprocessor computer organizations," IEEE Trans. Computers C-24, 11 (Nov. 1975), 1066-1074.

[Sea 75] Searle, B.C., and Freberg, D.E., "Tutorial : microprocessor applications in multiple processor systems," Computer 8,10 (Oct. 1975), 22-30.

[Ski 69] Skinner, C.E., and Asher, J.R., "Effects of storage contention on system performance," IBM Systems J. 8,4(1969), 319-333.

[Str 70] Strecker, W., "An analysis of the instruction execution rate in certain computer structures," Ph.D. dissertation, Carnegie-Mellon University, Pittsburgh, Pa., 1970.

[Swa 76] Swan, R.J., Fuller, S.H., and Siewiorek, D.P., "The structure and architecture of Cm* : a modular, multi-microprocessor," Computer Science Research Review 1975-76, Carnegie-Mellon Univ., Pittsburgh, Pa. (1976), 25-47.

[Thu 72] Thurber, K.J., Jensen, E.D., Jack, L.A., Kinney, L.L., Patton, P.C., and Anderson, L.C., "A systematic approach to the design of digital bussing structures," in 1972 Fall Joint Computer Conf., AFIPS Conf. Proc. Vol. 41. AFIPS Press, Montvale, N.J. (1972), 719-740.

[Wul 72] Wulf, W.A., and Bell, C.G., "C.mmp - a multimini processor," in 1972 Fall Joint Computer Conf., AFIPS Conf. Proc. Vol. 41. AFIPS Press, Montvale, N.J. (1972), 765-777.

APPENDIX

TRANSITION MATRIX FOR THE BUS MODEL EXAMPLE

For the bus model discussed in Section 4.4, with $p=2$, $n_1=2$, $n_2=1$, the transition matrix is:

$$
\Pi = \begin{bmatrix}
\alpha^2 & 0 & 2\alpha\beta & 0 & 0 & 0 & 0 & 0 & 0 & \beta^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^2 & 0 & 2\alpha\beta & 0 & 0 & 0 & 0 & 0 & 0 & \beta^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{2}{3}\alpha & \frac{1}{3}\alpha & 0 & 0 & 0 & 0 & 0 & \frac{2}{3}\beta & \frac{1}{3}\beta & 0 & 0 & 0 & 0 \\
0 & \alpha\delta_1 & 0 & \alpha\gamma_1 & 0 & 0 & \beta\delta_1 & 0 & 0 & 0 & \beta\gamma_1 & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha\delta_2 & 0 & 0 & \alpha\gamma_2 & 0 & \beta\delta_2 & 0 & 0 & 0 & 0 & \beta\gamma_2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha\delta_1 & 0 & \alpha\gamma_1 & 0 & 0 & \beta\delta_1 & 0 & 0 & 0 & \beta\gamma_1 & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha\delta_2 & 0 & 0 & \alpha\gamma_2 & 0 & \beta\delta_2 & 0 & 0 & 0 & 0 & \beta\gamma_2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{3}\delta_1 & \frac{1}{3}\delta_1 & 0 & 0 & 0 & \frac{1}{3}\gamma_1 & \frac{1}{3}\gamma_1 & \frac{1}{3}\gamma_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{3}\delta_2 & \frac{1}{3}\delta_2 & 0 & 0 & 0 & 0 & 0 & \frac{2}{3}\gamma_2 & \frac{1}{3}\gamma_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & 0 & 0 & 0 & 0 & \gamma_1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \delta_1^2 & 0 & 2\delta_1\gamma_1 & 0 & 0 & 0 & 0 & 0 & \gamma_1^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \delta_1\delta_2 & 0 & \gamma_1\delta_2 & \delta_1\gamma_2 & 0 & 0 & 0 & 0 & 0 & \gamma_1\gamma_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_2 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_2
\end{bmatrix}
$$